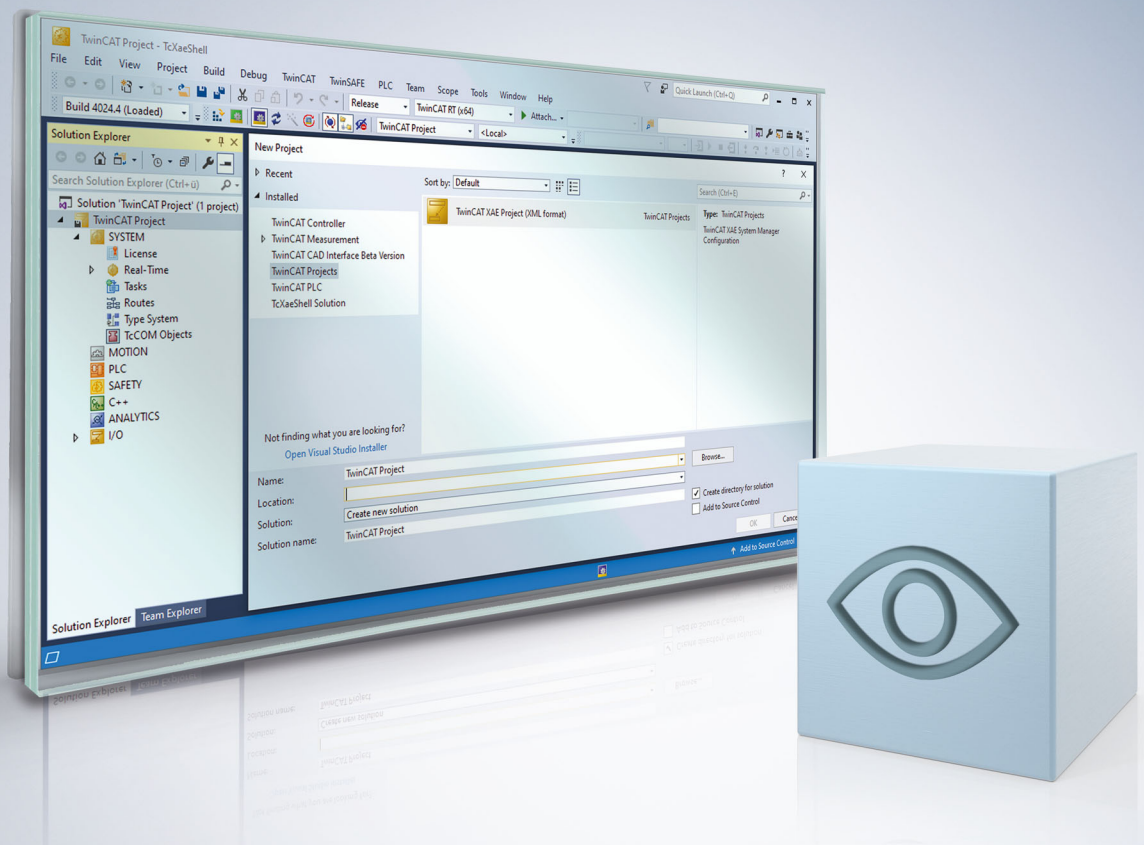
BECKHOFF New Automation Technology

Handbuch | DE

TF7000 - TF7810

TwinCAT 3 | Vision



15.05.2025 | Version: 2.3.1

Inhaltsverzeichnis

1 Vorwort ... 7
1.1 Hinweise zur Dokumentation ... 7
1.2 Zu Ihrer Sicherheit ... 8
1.3 Hinweise zur Informationssicherheit ... 9
1.4 Ausgabestände der Dokumentation ... 9

2 Übersicht ... 10
2.1 System-Komponenten ... 10
2.2 Lizenzmodell ... 12

3 Installation ... 15
3.1 Systemvoraussetzungen ... 15
3.2 Versionsübersicht ... 16
3.3 Installationsanleitung TwinCAT 3.1 Build 4024 ... 18
3.4 Installationsanleitung ab TwinCAT 3.1 Build 4026 ... 24
3.5 Verwendung der TwinCAT 3 Usermode Runtime ... 24
3.6 Lizenzierung ... 25

4 First Steps ... 28
4.1 Projekt erstellen ... 28
4.1.1 TwinCAT-Projekt erstellen ... 28
4.1.2 Zielsystem auswählen ... 30
4.1.3 System konfigurieren ... 32
4.2 Vision-Gerät einrichten ... 34
4.2.1 Vision-Knoten einblenden und Applikation erstellen ... 34
4.2.2 File Source Control anlegen ... 36
4.2.3 GigE Vision Kamera anlegen ... 39
4.3 SPS-Projekt erstellen ... 47
4.3.1 SPS-Projekt erstellen ... 47
4.3.2 SPS-Bibliothek einbinden ... 48
4.3.3 SPS-Programm schreiben ... 50
4.3.4 Funktionsbaustein initialisieren ... 51
4.4 Ausführen ... 52
4.4.1 TwinCAT-Projekt aktivieren und SPS-Projekt starten ... 52
4.4.2 Bildanzeige im ADS Image Watch ... 53

5 Entwicklungsumgebung ... 55
5.1 Systemkonfiguration ... 56
5.1.1 Router-Speicher ... 56
5.1.2 Core Memory ... 58
5.1.3 Stack-Größe ... 58
5.1.4 Netzwerkadapter ... 58
5.1.5 CPU-Kerne und Tasks ... 63
5.1.6 Job Tasks ... 66
5.2 VISION Knoten ... 68
5.2.1 Service Configuration ... 68
5.2.2 Logging ... 71

5.3 Application Knoten ... 71
5.4 GigE Vision Camera-Objekte ... 72
5.4.1 General ... 74
5.4.2 Configuration Assistant ... 79
5.4.3 Record/Playback ... 109
5.4.4 Camera Calibration ... 116
5.4.5 GigE Vision Standard ... 131
5.4.6 TcCOM-Objekte ... 132
5.5 Beckhoff Camera-Objekte ... 135
5.6 File Source Objekte ... 135
5.6.1 File Source Control ... 136
5.6.2 TcCOM-Objekte ... 139
5.7 Vision Job Pool ... 140
5.8 ADS Image Watch ... 141
5.9 Exportieren/Importieren von Knoten ... 144

6 API-Referenz ... 146
6.1 SPS ... 146
6.1.1 Softwarekonzept ... 146
6.1.2 Data Types ... 170
6.1.3 Interfaces ... 259
6.1.4 Functions ... 448
6.1.5 Function Blocks ... 1620
6.2 C++ ... 1710
6.2.1 Data Types ... 1710
6.2.2 Interfaces ... 1783
6.2.3 Functions ... 1944
6.2.4 Function Blocks ... 2732

7 TwinCAT HMI Pakete ... 2784
7.1 Server-Erweiterung ... 2784
7.2 Controls ... 2789
7.2.1 Image Watch ... 2790
7.2.2 Color ... 2815
7.2.3 Rectangle (Interactive) ... 2828
7.2.4 Polygon (Interactive) ... 2830
7.2.5 Hilfsfunktionen ... 2832
7.2.6 Datentypen ... 2835

8 Beispiele und Tools ... 2838
8.1 Beispiele zu Funktionen ... 2839
8.1.1 Basic Container Operations ... 2840
8.1.2 Basic Image Operations ... 2842
8.1.3 Code Reading ... 2845
8.1.4 Contour Analysis ... 2857
8.1.5 Image Analysis ... 2867
8.1.6 Image Color and Contrast Processing ... 2887
8.1.7 Image Filtering ... 2892

8.1.8 Image Segmentation ... 2897
8.1.9 Measurement ... 2900
8.1.10 Selbstgeschriebene Funktionen ... 2917
8.2 Beispiele zu Funktionsblöcken ... 2919
8.2.1 Camera Register Access ... 2919
8.2.2 File Access ... 2921
8.2.3 Image Acquisition ... 2923
8.3 Beispiele zu Assistenten ... 2925
8.3.1 Calibration Assistent ... 2925
8.4 Beispiele zur Kamera-Konfiguration ... 2932
8.4.1 Region of Interest (ROI) ... 2932
8.4.2 Binning ... 2933
8.4.3 Bild-Akquise und Trigger ... 2936
8.5 Neural Network Compatibility Check ... 2942

9 Appendix ... 2946
9.1 FAQ ... 2946
9.2 Wichtige Pfade ... 2947
9.3 Troubleshooting ... 2948
9.3.1 TwinCAT System Start ... 2948
9.3.2 SPS-Laufzeitumgebung ... 2952
9.3.3 Assistenten ... 2959
9.3.4 TwinCAT HMI Pakete ... 2963
9.3.5 Kamerakommunikation ... 2964
9.4 ADS Return Codes ... 2966
9.5 Support und Service ... 2971
9.6 Third-party Components ... 2973

1 Vorwort

1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.
Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.
Das Fachpersonal ist verpflichtet, stets die aktuell gültige Dokumentation zu verwenden.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbaren Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiterentwickelt.
Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.
Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Marken

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® und XPlanar® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.
Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.

EtherCAT® ist eine eingetragene Marke und patentierte Technologie lizenziert durch die Beckhoff Automation GmbH, Deutschland

Copyright

© Beckhoff Automation GmbH & Co. KG, Deutschland.
Weitergabe sowie Vervielfältigung dieses Dokuments, Verwertung und Mitteilung seines Inhalts sind verboten, soweit nicht ausdrücklich gestattet.
Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patent-, Gebrauchsmuster- oder Geschmacksmustereintragung vorbehalten.

Fremdmarken

In dieser Dokumentation können Marken Dritter verwendet werden. Die zugehörigen Markenvermerke finden Sie unter: https://www.beckhoff.com/trademarks.

1.2 Zu Ihrer Sicherheit

Sicherheitsbestimmungen

Lesen Sie die folgenden Erklärungen zu Ihrer Sicherheit.
Beachten und befolgen Sie stets produktspezifische Sicherheitshinweise, die Sie gegebenenfalls an den entsprechenden Stellen in diesem Dokument vorfinden.

Haftungsausschluss

Die gesamten Komponenten werden je nach Anwendungsbestimmungen in bestimmten Hard- und Software-Konfigurationen ausgeliefert. Änderungen der Hard- oder Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig und bewirken den Haftungsausschluss der Beckhoff Automation GmbH & Co. KG.

Qualifikation des Personals

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen vertraut ist.

Signalwörter

Im Folgenden werden die Signalwörter eingeordnet, die in der Dokumentation verwendet werden. Um Personen- und Sachschäden zu vermeiden, lesen und befolgen Sie die Sicherheits- und Warnhinweise.

Warnungen vor Personenschäden

⚠ GEFAHR

Es besteht eine Gefährdung mit hohem Risikograd, die den Tod oder eine schwere Verletzung zur Folge hat.

⚠ WARNUNG

Es besteht eine Gefährdung mit mittlerem Risikograd, die den Tod oder eine schwere Verletzung zur Folge haben kann.

⚠ VORSICHT

Es besteht eine Gefährdung mit geringem Risikograd, die eine mittelschwere oder leichte Verletzung zur Folge haben kann.

Warnung vor Umwelt- oder Sachschäden

HINWEIS

Es besteht eine mögliche Schädigung für Umwelt, Geräte oder Daten.

Information zum Umgang mit dem Produkt

Diese Information beinhaltet z. B.:
Handlungsempfehlungen, Hilfestellungen oder weiterführende Informationen zum Produkt.

1.3 Hinweise zur Informationssicherheit

Die Produkte der Beckhoff Automation GmbH & Co. KG (Beckhoff) sind, sofern sie online zu erreichen sind, mit Security-Funktionen ausgestattet, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen. Trotz der Security-Funktionen sind die Erstellung, Implementierung und ständige Aktualisierung eines ganzheitlichen Security-Konzepts für den Betrieb notwendig, um die jeweilige Anlage, das System, die Maschine und die Netzwerke gegen Cyber-Bedrohungen zu schützen. Die von Beckhoff verkauften Produkte bilden dabei nur einen Teil des gesamtheitlichen Security-Konzepts. Der Kunde ist dafür verantwortlich, dass unbefugte Zugriffe durch Dritte auf seine Anlagen, Systeme, Maschinen und Netzwerke verhindert werden. Letztere sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn entsprechende Schutzmaßnahmen eingerichtet wurden.

Zusätzlich sollten die Empfehlungen von Beckhoff zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Informationssicherheit und Industrial Security finden Sie in unserem https://www.beckhoff.de/secguide.

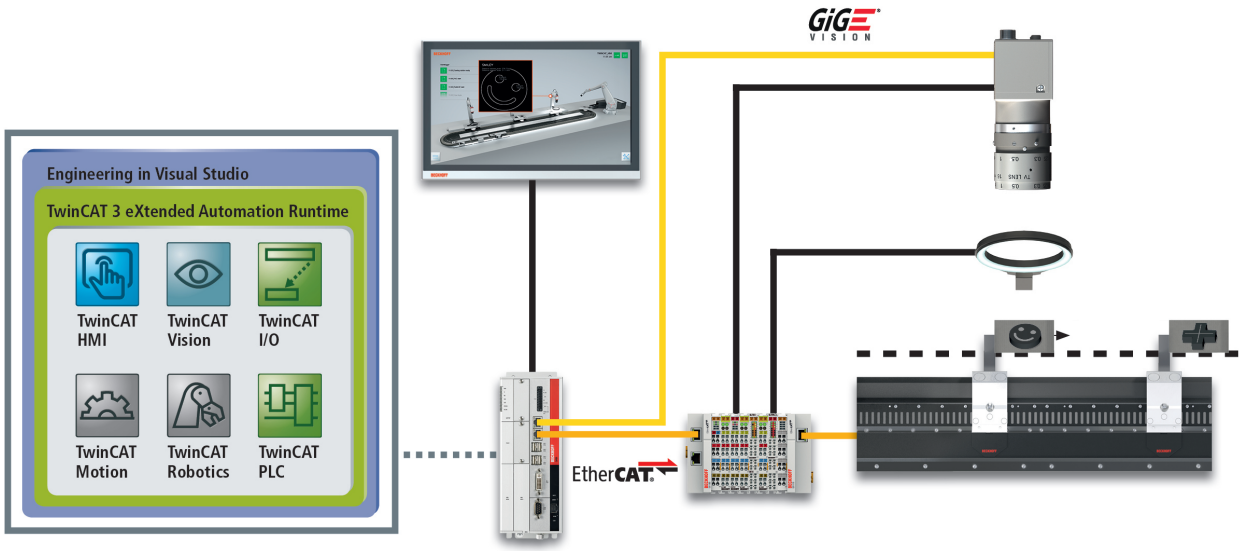
Die Produkte und Lösungen von Beckhoff werden ständig weiterentwickelt. Dies betrifft auch die Security-Funktionen. Aufgrund der stetigen Weiterentwicklung empfiehlt Beckhoff ausdrücklich, die Produkte ständig auf dem aktuellen Stand zu halten und nach Bereitstellung von Updates diese auf die Produkte aufzuspielen. Die Verwendung veralteter oder nicht mehr unterstützter Produktversionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Hinweise zur Informationssicherheit zu Produkten von Beckhoff informiert zu sein, abonnieren Sie den RSS Feed unter https://www.beckhoff.de/secinfo.

1.4 Ausgabestände der Dokumentation

Version	Änderung
2.3.x	**Neu:** diverse Datentypen und Funktionen Installation/Verwendung der TwinCAT 3 Usermode Runtime [▶ 24]
2.2.x	**Neu:** Neural Network und Neural Network C++
2.1.x	**Neu:** diverse Funktionen, Funktionen Code Quality [▶ 843] und Code Quality C++ [▶ 2258] **Neu:** Installationsanleitung für TwinCAT 3.1 Build 4026 [▶ 24]
2.0.x	**Neu:** diverse Funktionsbausteine, Funktionen, Datentypen und Schnittstellen [▶ 146] **Neu:** API für C++ [▶ 1710]
1.5.x	**Neu:** diverse Funktionsbausteine und Funktionen

2 Übersicht



TwinCAT Vision ist eine TwinCAT 3 Function für industrielle Bildverarbeitung. Aufgaben wie die Detektion, Identifikation oder Vermessung von Objekten können direkt in der SPS in Echtzeit gelöst werden. Durch die Integration der Bildverarbeitung in die TwinCAT-Plattform sind hoch-synchronisierte Steuerungsabläufe und kürzeste Reaktionszeiten realisierbar. Die Applikationsentwicklung wird durch die Vielzahl von bereits vorhandenen TwinCAT Engineering-Tools unterstützt.

Die Programmierung von Bildverarbeitungsabläufen erfolgt in den IEC61131-3 Sprachen mit Hilfe der Tc3_Vision Bibliothek direkt im SPS-Code. Alternativ kann die Programmierung auch in TwinCAT 3 C++ erfolgen. Zwischenergebnisse können durch ein Bildanzeigetool direkt überprüft werden.

TwinCAT Vision ist Echtzeit-fähig, da die Bildverarbeitungsalgorithmen synchron zur Steuerung direkt in der TwinCAT-3-Laufzeitumgebung ausgeführt werden. Geeignete Algorithmen können zudem automatisch auf mehreren Rechenkernen parallelisiert ausgeführt werden.

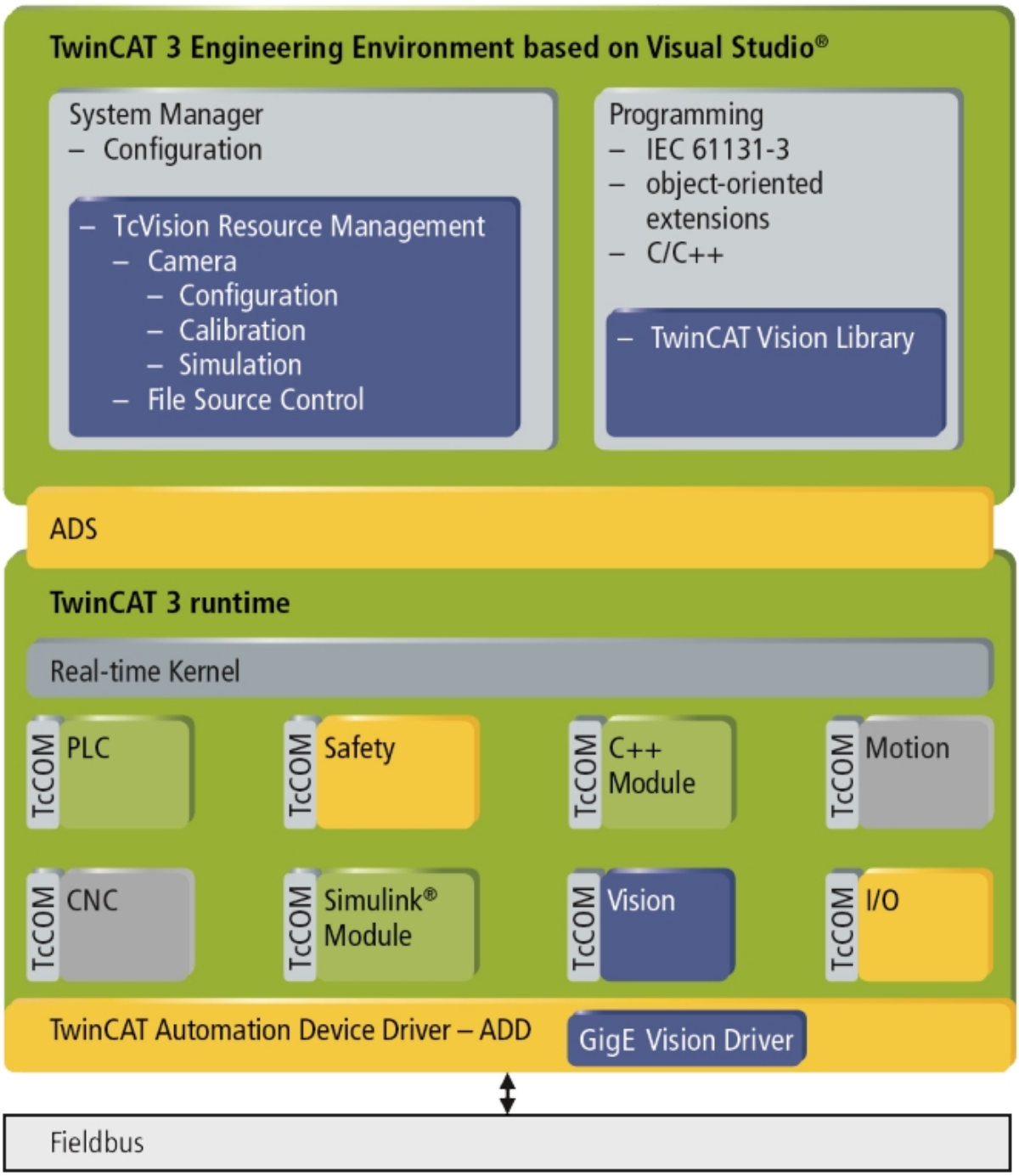
Über die GigE Vision Schnittstelle können zahlreiche Industriekameras angebunden werden. Die bekannte TwinCAT-Entwicklungsumgebung wird zum Konfigurieren und Kalibrieren von Kameras sowie zur Systemeinrichtung und zum Debugging genutzt.

In den weiteren Kapiteln finden Sie ausführliche Informationen zu den Themen:

- System-Komponenten [▶ 10] zum Verständnis des Systemaufbaus von TwinCAT Vision
- Installation [▶ 15] für Anleitungen zur Installation und Lizensierung
- First Steps [▶ 28] für einen schnellen Einstieg in TwinCAT Vision
- Entwicklungsumgebung [▶ 55] zur Konfiguration, Kalibrierung und Simulation von Kameras
- API-Referenz [▶ 146] (Software-Dokumentation)
- Beispiele [▶ 2838] einiger Bildverarbeitungsabläufe
- Hinweise [▶ 2948] zum Troubleshooting

2.1 System-Komponenten

Analog zu anderen TwinCAT Komponenten ist TwinCAT Vision in die auf Visual Studio basierende TwinCAT-Entwicklungsumgebung und in die Echtzeit-fähige TwinCAT-Laufzeitumgebung integriert. Folgend sind die Bestandteile des TwinCAT Vision Systems erklärt.



Konfigurationsassistenten

- Industriekameras können über die GigE Vision [▶ 131] Schnittstelle angebunden werden. Dabei stehen Assistenten [▶ 72] zur Konfiguration, Kalibrierung und Simulation der Kameras zur Verfügung.
- Die Einspeisung von Bildern aus dem Dateisystem in die SPS Laufzeit ist durch ein sog. File Source [▶ 135] Element möglich.
- Die Konfiguration [▶ 56] von Rechenkernen und dem Router-Speicher für TwinCAT Vision ist grundsätzlich in den TwinCAT Echtzeiteinstellungen vorzunehmen.
- TwinCAT Vision kann TwinCATs Multi-Core-Unterstützung nutzen, um bestimmte Algorithmen parallelisiert auszuführen. Entsprechende Einstellungen erfolgen im Vision Job Pool [▶ 140].

Bibliothek

Zur Programmierung von Bildverarbeitungsabläufen stellt TwinCAT Vision eine Bibliothek mit einer Vielzahl von Algorithmen bereit. Darin sind ebenfalls Funktionsblöcke enthalten, die als Schnittstelle für Kameras und das Dateisystem dienen. Als detaillierte Beschreibung aller Bibliothekselemente existiert die API-Referenz [▶ 146].

Debugging-Werkzeuge

Zum Debugging von Bildverarbeitungsabläufen stellt TwinCAT Vision folgende Mechanismen zur Verfügung:

- Das ADS Image Watch [▶ 141] dient der Live-Visualisierung von Kamera-Aufnahmen und Zwischenergebnissen. Es überträgt Bilder aus der SPS-Laufzeit per ADS in die Entwicklungsumgebung und zeigt sie dort an.
- Sämtliche TwinCAT Vision Funktionen geben basierend auf ADS Return Codes [▶ 2966] eine Auskunft über ihren Ausführungsstatus (siehe HRESULT [▶ 150]).
- Für die Programmierung können sämtliche TwinCAT-Debugging Funktionen wie Breakpoints u. Ä. genutzt werden.

GigE Vision Treiber

TwinCAT Vision enthält einen Echtzeit-fähigen GigE Vision [▶ 131] Treiber. Er wird sowohl in der Entwicklungsumgebung zur Kamera-Konfiguration, als auch in der TwinCAT Laufzeit genutzt. Er kommuniziert gemäß dem GigE Vision Standard über UDP/IP mit Industriekameras. Über diese Schnittstelle werden Kameras im Netzwerk gefunden und konfiguriert sowie Bilddaten von der Kamera empfangen.

TwinCAT Vision Service

Für einige Funktionalitäten muss TwinCAT Vision zur Laufzeit mit Komponenten außerhalb der Echtzeit kommunizieren. Beispielsweise lädt das File Source Control Bilder aus dem Dateisystem ein. Dazu stellt der TwinCAT Vision Service [▶ 68] eine zentralisierte, auf ADS basierende Kommunikationsschnittstelle zwischen der TwinCAT-Laufzeit und externen Vision-Komponenten dar.

TwinCAT HMI Vision Extension

Die TwinCAT HMI Vision Extension [▶ 2784] ist eine Erweiterung für die TwinCAT HMI, die die Übertragung von Bildern aus der SPS hin zur TwinCAT HMI ermöglicht. So können Bilder aus der SPS direkt mit dem HmiImageControl in der Web-basierten HMI angezeigt werden.

TwinCAT HMI Vision Controls

Das TwinCAT HMI Vision Controls [▶ 2789] Paket stellt Vision spezifische Controls als Erweiterung für die TwinCAT HMI bereit. Es beinhaltet z. B. ein Control, das zusätzlich zur Bildanzeige noch viele weitere Werkzeuge und Informationsanzeigen bereitstellt. Die Funktionen und Anzeigen können per Konfiguration einfach angepasst und ohne zusätzliche Programmierung direkt verwendet werden.

2.2 Lizenzmodell

Die TF7xxx-Produkte für TwinCAT Vision teilen sich auf in Lizenzen für Kameraanbindungen und für eine Bildverarbeitungsbibliothek. Für das Generieren von Lizenzen folgen Sie bitte dieser Anleitung [▶ 25]. Zum Testen oder Entwickeln mit TwinCAT Vision können Sie Lizenzen für eine 7-Tage-Testversion erstellen.

Die aktuell verfügbaren Produktlizenzen entnehmen Sie bitte der Produkt-Website.

TF700x | TwinCAT 3 GigE Vision Connector

Der TwinCAT 3 GigE Vision Connector ermöglicht sowohl die Anbindung von GigE Vision Kameras als auch von Beckhoff Kameras. Die Basis-Lizenz TF7000 wird immer benötigt und ermöglicht die Anbindung von zwei Kameras. Bei Bedarf kann die Kapazität mit Erweiterungslizenzen erhöht werden:

TF7000	Basis-Lizenz inkl. 2 Kamera-Verbindungen (wird immer benötigt)
TF7001	Erweiterung für 2 zusätzliche Verbindungen
TF7002	Erweiterung für 4 zusätzliche Verbindungen
TF7003	Erweiterung für 8 zusätzliche Verbindungen

Die Erweiterungslizenzen können miteinander kombiniert werden, sodass maximal 2+2+4+8=16 Kameras verbunden werden können.

Zum Beispiel werden die folgenden Lizenzen für 11 Kameraanbindungen benötigt:

- TF7000 Base (inkl. 2 Kameraanbindungen)
- TF7001 2 Camera
- TF7003 8 Camera

→ In Summe können nun 12 Kameras angebunden werden.

TF7020 | TwinCAT 3 Vision Beckhoff Camera Connector

Der TwinCAT 3 Vision Beckhoff Camera Connector ermöglicht die Anbindung von bis zu 64 Beckhoff-Kameras. Kameras von anderen Herstellern können mit dieser Lizenz nicht verwendet werden.

TF7100-TF7810 | TwinCAT 3 Vision Algorithms

Die Produktlizenzen TF7100-TF7810 stellen Bildverarbeitungsfunktionen in der Bibliothek „Tc3_Vision“ zur Verfügung. Die Basislizenz TF7100 wird für jede Nutzung der Bibliothek benötigt und gewährt Zugriff auf grundlegenden Algorithmen wie Filteroperationen, Format-Konvertierungen und Konturverfolgung. Hiermit können schon viele Applikationen realisiert werden. Die Erweiterungslizenzen ergänzen den Funktionsumfang wie in der Tabelle beschrieben. Die Zugehörigkeit von Funktionen zu Lizenzen finden Sie in der API-Referenz [▶ 450].

TF7100	Basis-Lizenz für die „Tc3_Vision“-Bibliothek. Enthält Algorithmen zur Lösung von Bildverarbeitungsaufgaben, wie algebraische Operationen, Filter, Fourier-Analyse, Farbbildverarbeitung, Segmentierung, Kontur- und Blobanalyse, Ergebnisdarstellung sowie zum Lesen und Schreiben von Kameraparametern.
TF7200	Erweiterung für Matching 2D: Objekte können basierend auf eingelernten Referenzen, Konturen, Merkmalspunkten oder anderen Eigenschaften gefunden und verglichen werden (Template Matching / KeyPoint Detektion und Deskriptor Matching). Eine häufige Anwendung ist die Klassifikation von Objekten.
TF7250	Erweiterung für Code Reading: Detektion und Auslesen von 1D- und 2D-Codes.
TF7255	Erweiterung für Code Quality: Qualitätsbewertung von 1D- und 2D-Codes.
TF7260	Erweiterung für OCR: Funktionen zur optischen Zeichenerkennung, die Zeichen in einer Bildregion identifizieren und die erkannte Zeichenkette zurückgeben.
TF7300	Erweiterung für Metrology 2D: Tools zur optischen Vermessung von geometrischen Objekteigenschaften (Abstände, Radien, etc.). • Subpixel-genaue Kantenlokalisierung auf den Messbildern • Kompensation von optischen Verzeichnungen (vorherige Kalibrierung der Kamera notwendig!) • Transformation zwischen Pixel-Koordinaten und Echtwelt-Koordinaten (vorherige Kalibrierung der Kamera notwendig!)
TF7800	Erweiterung für Machine Learning: Funktionen zum Training und der Ausführung von verschiedenen klassischen ML-Modellen.
TF7810	Erweiterung für Neural Network: Funktionen zur Ausführung von trainierten neuronalen Netzen im Open Neural Network Exchange (ONNX) Format.

Automatische Ermittlung von erforderlichen Lizenzen der Vision Bibliothek

Die Ermittlung der erforderlichen Lizenzen erfolgt erst beim Kompilieren des jeweiligen TwinCAT Projekts. Dabei wird geschaut, welche Funktionsbausteine deklariert und welche Funktionen im Code verwendet werden.

Achten Sie daher darauf, dass bei (kurzfristigen) Projektänderungen benötigte Lizenzen hinzukommen oder wegfallen können. Wenn neue Lizenzen hinzukommen, muss die Änderung durch ein **Activate Configuration** eingespielt werden, ein Online-Change ist in dem Fall nicht möglich.

3 Installation

Folgend finden Sie die Systemanforderungen [▶ 15] für TwinCAT Vision, die Versionsübersicht [▶ 16], sowie Anleitungen zur Installation [▶ 18] und Lizensierung [▶ 25].

3.1 Systemvoraussetzungen

IPC Hardware

Technische Daten	Beschreibung
TwinCAT 3 Plattform-Level	Minimum P50 Performance Plus, z. B. Intel 4-Kern-Atom CPU
Netzwerkadapter	TwinCAT Realtime Ethernet-kompatible Gigabit-Netzwerkkarten (zum Anschluss von GigE Vision Kameras) **Mindestens ein Netzwerkport sollte ausschließlich für Kameras zur Verfügung stehen. Empfohlen wird die Verwendung eines separaten Netzwerkports pro Kamera.**
Arbeitsspeicher	Minimum 4 GB empfohlen für reine Runtime-Systeme Minimum 8 GB empfohlen für Entwicklungs-Systeme Minimum 8 GB empfohlen bei der Verwendung von Record/Playback [▶ 109] Streams auf Runtime-Systemen
Festplatte	Empfohlen SSD-Festplatten mit hohen Lese-/Schreibraten, besonders wenn viele Daten zwischen SPS und Festplatte ausgetauscht werden sollen.

Kamera

Technische Daten	Beschreibung
Schnittstelle	Muss den GigE Vision Standard [▶ 131] unterstützen. (Kennzeichnung üblicherweise durch GigE Vision Logo) **Der TwinCAT GigE Vision Connector ist nach dem GigE Vision Standard Version 2.0 zertifiziert. Dies bedeutet aber nicht, dass alle Kamera- oder GigE Vision Features der Version unterstützt werden.** Bei der Verwendung einer Kamera in der TwinCAT 3 Usermode Runtime sind die im Kapitel Verwendung der TwinCAT 3 Usermode Runtime [▶ 24] beschriebenen Punkte zu beachten.

Software

Technische Daten	Beschreibung
Betriebssystem	Windows 10 (nur 64 Bit!) Windows 11 ab TwinCAT 3.1 Build 4026
Unterstützte Visual Studio Versionen	2017 (Community, Professional) 2019 (Community, Professional) 2022 (Community, Professional) ab TwinCAT 3.1 Build 4026 TwinCAT XAE Shell (TcXaeShell) TwinCAT XAE Shell (TcXaeShell64) ab TwinCAT 3.1 Build 4026
TwinCAT	Minimal Version 3.1.4024.59 Minimal Version 3.1.4026.11
.NET Framework	Minimal Version 4.7.2 (nur für die Entwicklungsumgebung [▶ 55])

TwinCAT HMI Server-Erweiterung

Technische Daten	Beschreibung
.NET	Minimal Version .NET Desktop Runtime 8.0

3.2 Versionsübersicht

TwinCAT 3.1.4024 Setup	TwinCAT 3.1.4026 Workload	Bemerkung
4.0.7.4	5.7.4	
4.0.6.5	5.6.5	Neural Network hinzugefügt.
4.0.5.10	5.5.10	Beckhoff Camera Connector und Code Quality hinzugefügt.
4.0.4.8	-	C++ API, OCR-Modelle und Machine Learnig hinzugefügt.
4.0.3.5	-	Automatische Prüfung der Initialization Commands beim Übersetzungsprozess oder Aktivieren der Konfiguration wurde entfernt.
4.0.2.13	-	
4.0.1.3	-	Repository Dateien werden mit Installation eines neueren Setups gelöscht. **Wenn Sie diese Version beibehalten wollen, sichern Sie vor der Installation eines neuen Setups den „%TWINCAT3DIR% \3.1\Repository\Beckhoff Automation GmbH“ Ordner und kopieren ihn nach der Installation zurück.**

HINWEIS

Kompatibilität

Wenn Sie mehrere TwinCAT Vision Versionen auf einem System installiert haben oder sich Entwicklungs- und Laufzeitumgebung auf unterschiedlichen Systemen befinden, müssen Sie sicherstellen, dass Sie alle Komponenten mit der gleichen Version verwenden.

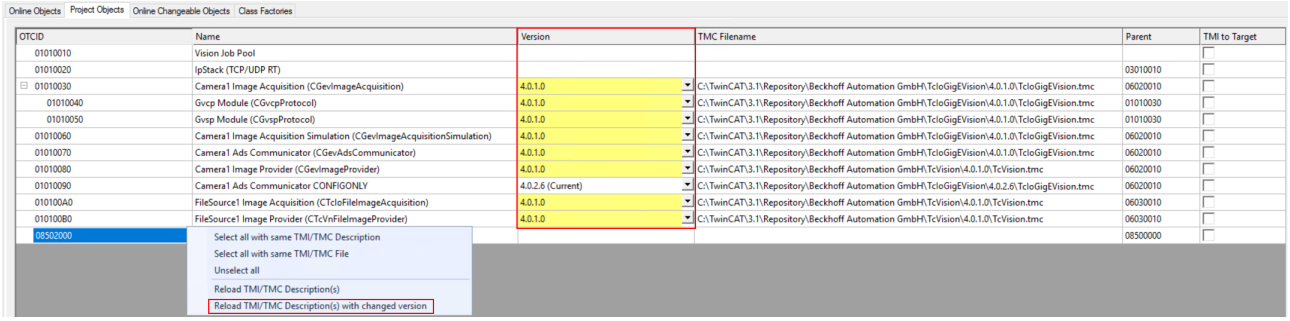
Der TwinCAT Vision Service und die Engineering Extensions liegen jeweils nur in der neusten Version vor. Daher bestimmen diese Komponenten die zu verwendende TcCOM- und Bibliotheksversion. Ein Vermischen von Versionen ist nicht zulässig und kann zu undefiniertem Fehlverhalten führen.

Hinweise für die Auswahl einer Version

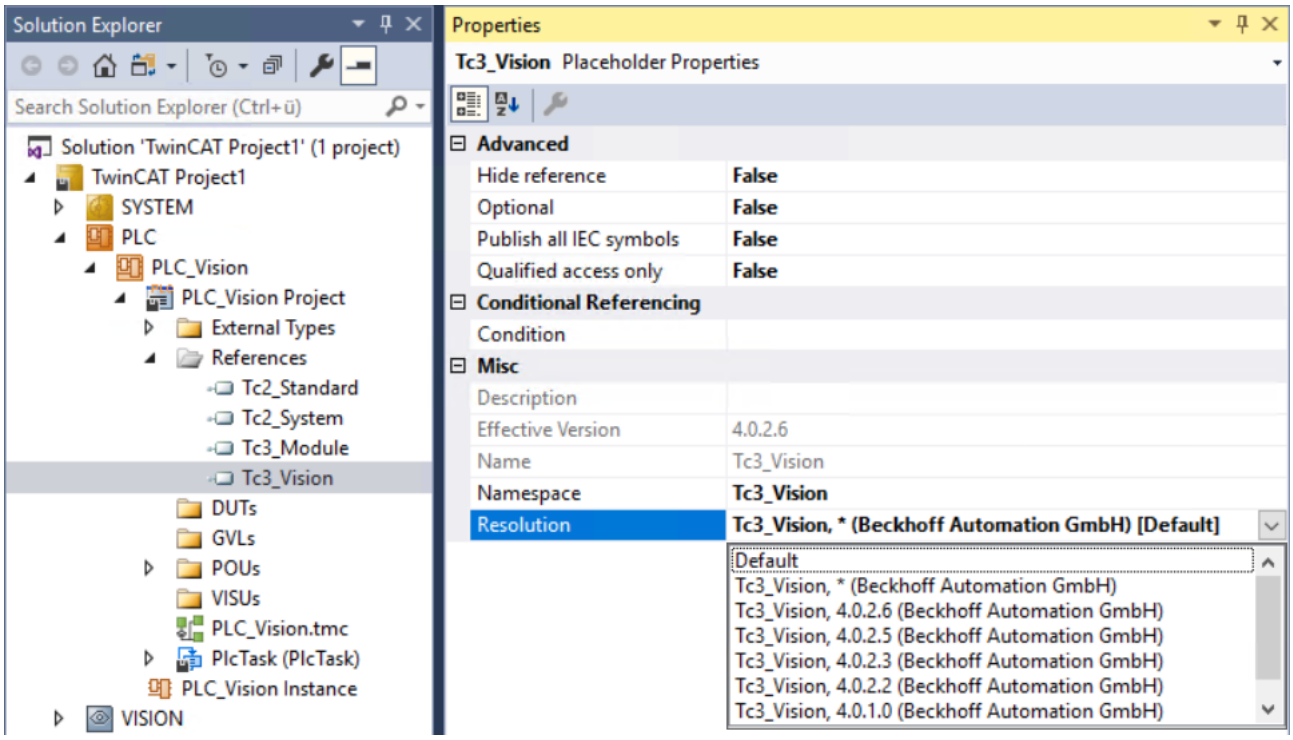
Wenn Sie TwinCAT Vision mit einer bestimmten Version verwenden wollen, müssen alle TcCOM-Objekte des Projekts auf die entsprechende Version aktualisiert werden. Eine Ausnahme stellt der Ads Communicator CONFIGONLY dar. Dieses Modul wird nur intern im Configuration Assistant verwendet und muss immer die neuste Version haben, die zur Extension passt. Bei diesem Modul wird daher beim Laden eines Projekts automatisch die neuste Version gesetzt.

Zum Prüfen oder Ändern der TcCOM Versionen:

1. Öffnen Sie das Fenster **SYSTEM > TcCOM Objects > Project Objects**.
2. Zum Ändern wählen Sie die gewünschte Version für alle Objekte aus.
3. Rechtsklicken Sie auf eine der Objekt-IDs in der ersten Spalte.
4. Wählen Sie **Reload TMI/TMC Description(s) with changed version**.



Weiterhin muss auch die gleiche Version der Tc3_Vision Bibliothek, entsprechend der ausgewählten TcCOM-Objekt Version, im SPS Projekt verwendet werden. Überprüfen Sie daher in den Eigenschaften der Tc3_Vision Bibliothek welche Version effektiv verwendet wird oder passen Sie diese entsprechend an.



Die aktuelle Versionsnummer des TwinCAT Vision Service und damit auch der Engineering Extensions finden Sie in der Service Configuration [▶ 68] unter dem Vision Knoten.

HINWEIS

Online-Change nach Update

Wenn Sie TwinCAT Vision auf einem Entwicklungssystem updaten und damit einen Online-Change auf einem Zielsystem mit einer älteren Version machen, kann dies zu Versionskonflikten bei Interface Pointern (Bildern und Containern) führen.

Löschen von älteren Versionen

Zum Löschen von älteren Versionen die mit einer 4.x.x.x beginnen sind zwei Schritte notwendig:

1. Zum einen müssen Sie die entsprechenden Versionsordner in den Verzeichnissen Tc3_Vision, TcIoGigEVision und TcVision unter „%TWINCAT3DIR%\3.1\Repository\Beckhoff Automation GmbH“ oder unter „C:\ProgramData\Beckhoff\TwinCAT\3.1\Repository\Beckhoff Automation GmbH“ löschen.
2. Um die entsprechenden SPS-Bibliotheksversionen zu entfernen, müssen Sie diese zum anderen noch über den Bibliotheksrepository Dialog deinstallieren.

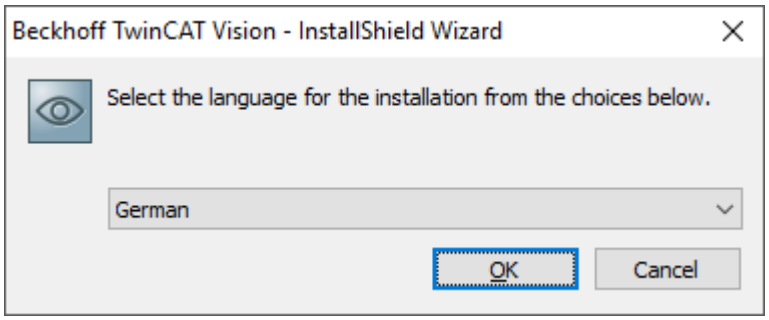
Zum Löschen von älteren Versionen die mit einer 5.x.x beginnen müssen Sie die entsprechenden Pakete mit Angabe der Version über die folgenden Kommandos deinstallieren.

- Zum Deinstallieren vom TcIoGigEVision Repository:
  ```
  tcpkg uninstall TwinCAT.XAE.TMX.GigEVision=5.x.x
  ```
- Zum Deinstallieren von den Tc3_Vision und TcVision Repositories:
  ```
  tcpkg uninstall TwinCAT.XAE.TMX.Vision=5.x.x
  ```
- Zum Deinstallieren von den klassischen OCR-Modellen:
  ```
  tcpkg uninstall TwinCAT.XAR.OcrModelClassic=5.x.x
  ```
- Zum Deinstallieren von den gemeinsam verwendeten Ressourcen, abschließend muss einmalig ein `repair` durchgeführt werden:
  ```
  tcpkg uninstall TwinCAT.XAE.VisionCommon=5.x.x
  tcpkg repair TwinCAT.XAE.VisionCommon
  ```

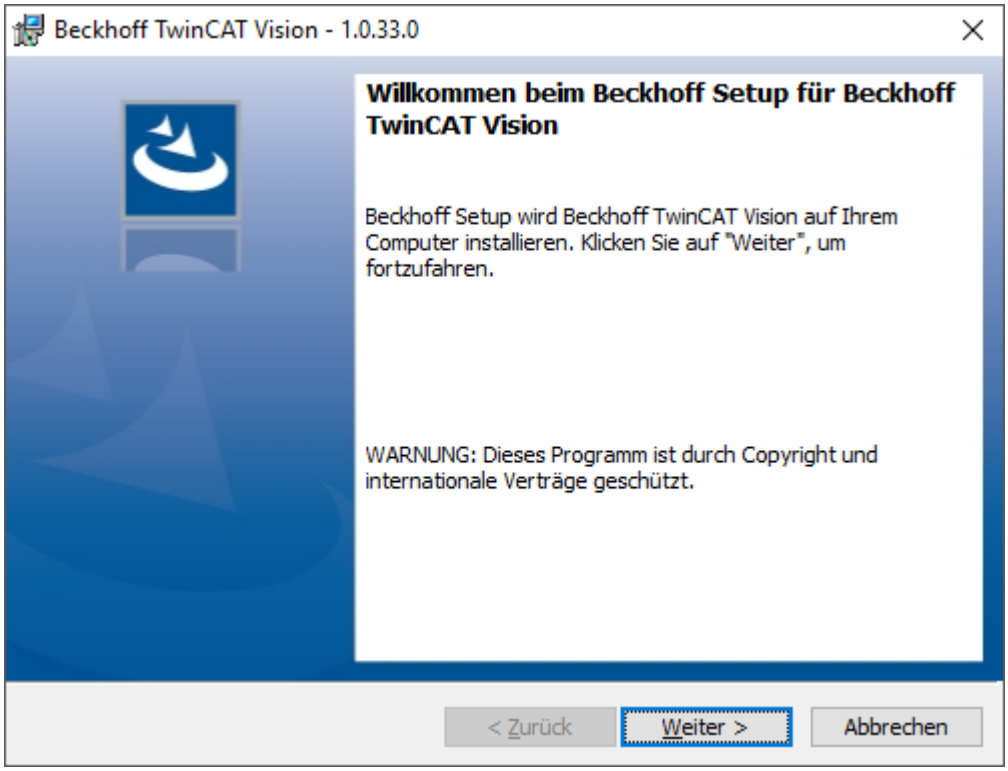
3.3 Installationsanleitung TwinCAT 3.1 Build 4024

Die Installation von TwinCAT Vision auf einem Windows-Betriebssystem besteht aus folgenden Schritten:

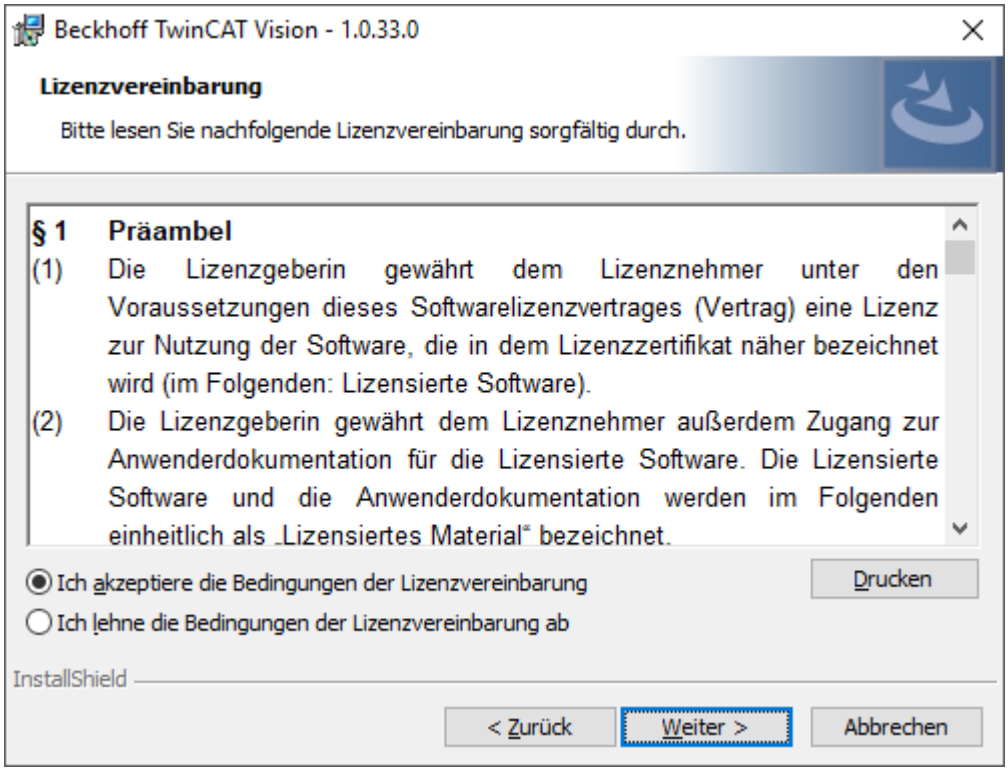
1. Stellen Sie sicher, dass sich TwinCAT im Config-Modus befindet und dass keine Visual Studio Instanz bzw. TcXaeShell geöffnet ist.
2. Führen Sie die Installationsdatei TF7xxx-Vision.exe als Administrator aus. Klicken Sie dazu mit der rechten Maustaste auf die Datei und wählen **Als Administrator ausführen**. Bestätigen Sie dies gegebenenfalls mit einem Klick auf **Ja**.
3. Wählen Sie die Sprache, in der die Installation vorgenommen werden soll. Klicken Sie dann auf **OK**.



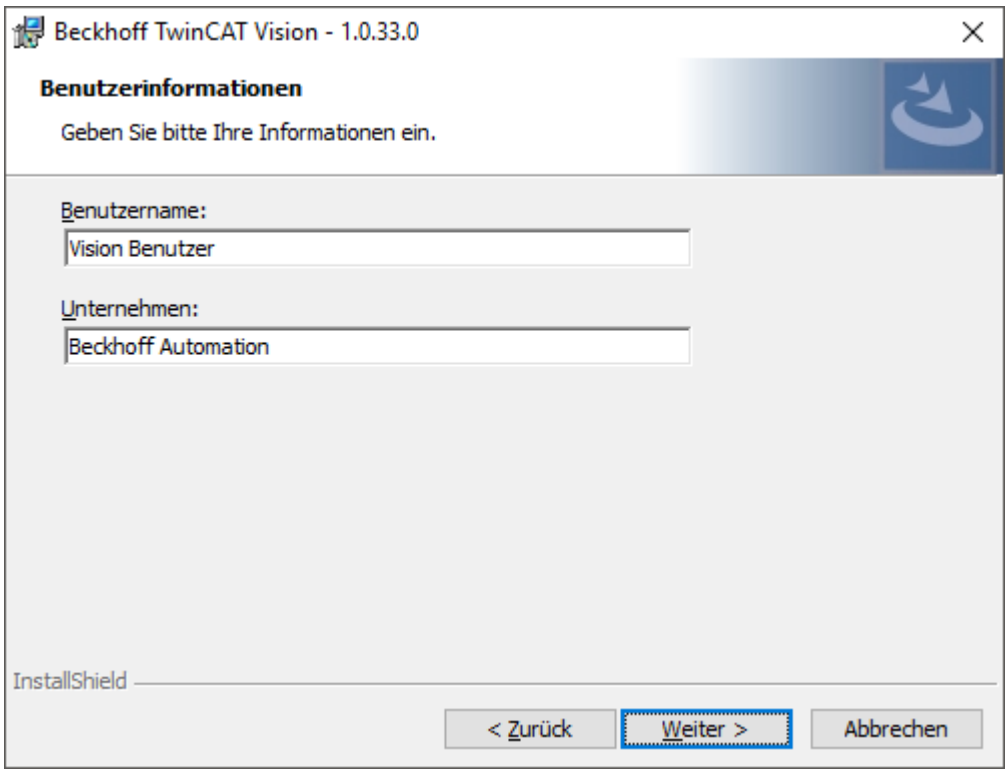
4. Warten Sie einen kurzen Moment, während die Installation vorbereitet wird und klicken Sie dann auf **Weiter**.



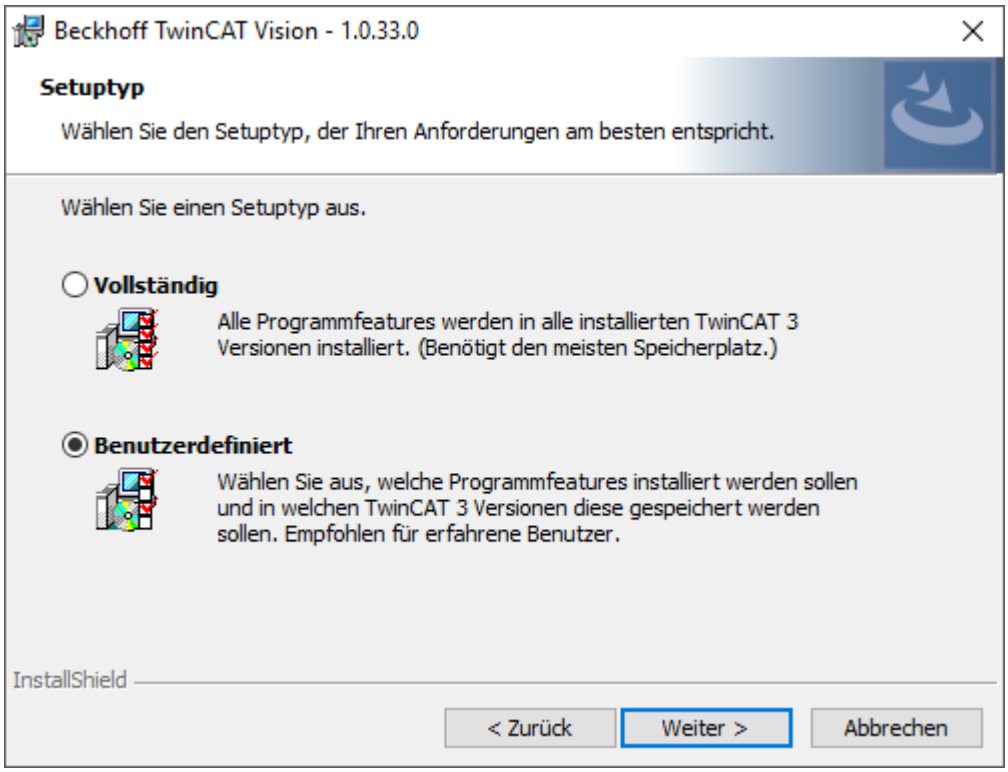
5. Stellen Sie sicher, dass Sie der Lizenzvereinbarung zustimmen und wählen Sie die entsprechende Option **Ich akzeptiere die Bedingungen der Lizenzvereinbarung**. Optional können Sie die Lizenzvereinbarung **Drucken**. Klicken Sie dann auf **Weiter**.



6. Geben Sie als Benutzerinformationen Ihren Benutzernamen und Ihr Unternehmen an. Klicken Sie dann auf **Weiter**.

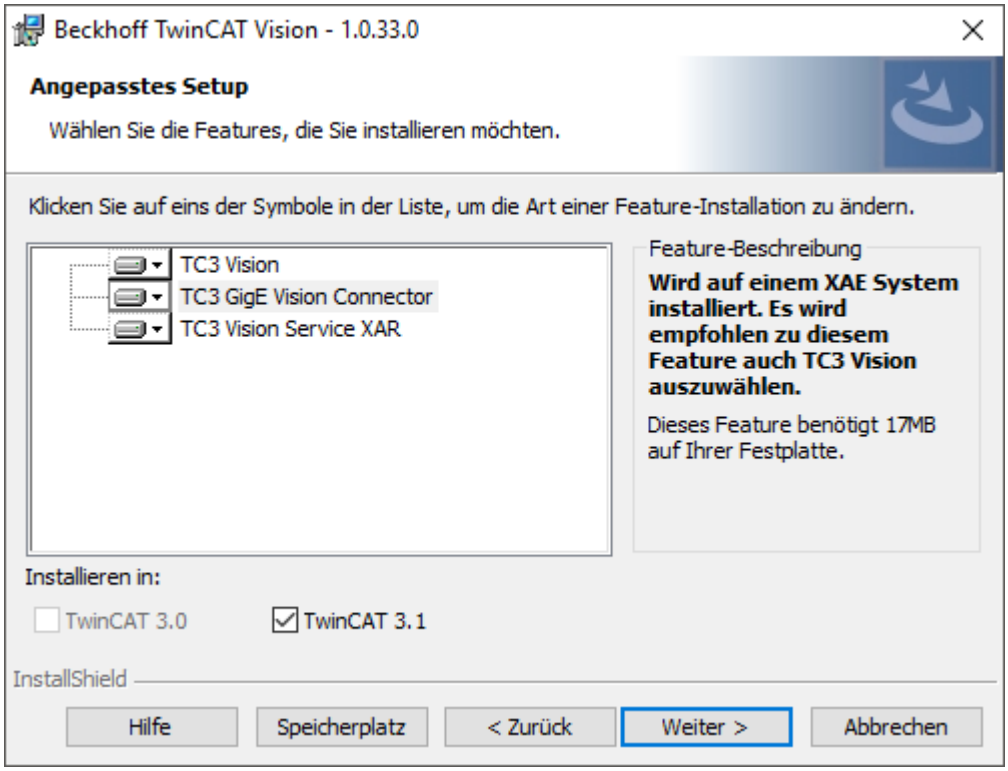


7. Wählen Sie zwischen den beiden Setuptypen **Vollständig** und **Benutzerdefiniert** aus. Klicken Sie dann auf **Weiter**.



- Ein vollständiges Setup installiert alle TwinCAT Vision Komponenten und benötigt den meisten Speicherplatz. Diese Option wird für XAE-Systeme empfohlen. Da auf reinen Runtime-Systemen nur der Vision Service installiert werden kann und die anderen Optionen nicht zur Auswahl stehen, empfiehlt sich diese Option hier ebenfalls.
- Ein benutzerdefiniertes Setup lässt Sie im nächsten Schritt wählen, welche TwinCAT Vision Komponenten Sie installieren möchten.

8. Wählen Sie für jede der folgenden TwinCAT Vision Komponenten, ob Sie diese installieren möchten. Wählen Sie zudem mindestens eine TwinCAT 3 Version aus, in der die auswählten Komponenten installiert werden sollen. Klicken Sie dann auf **Weiter**.

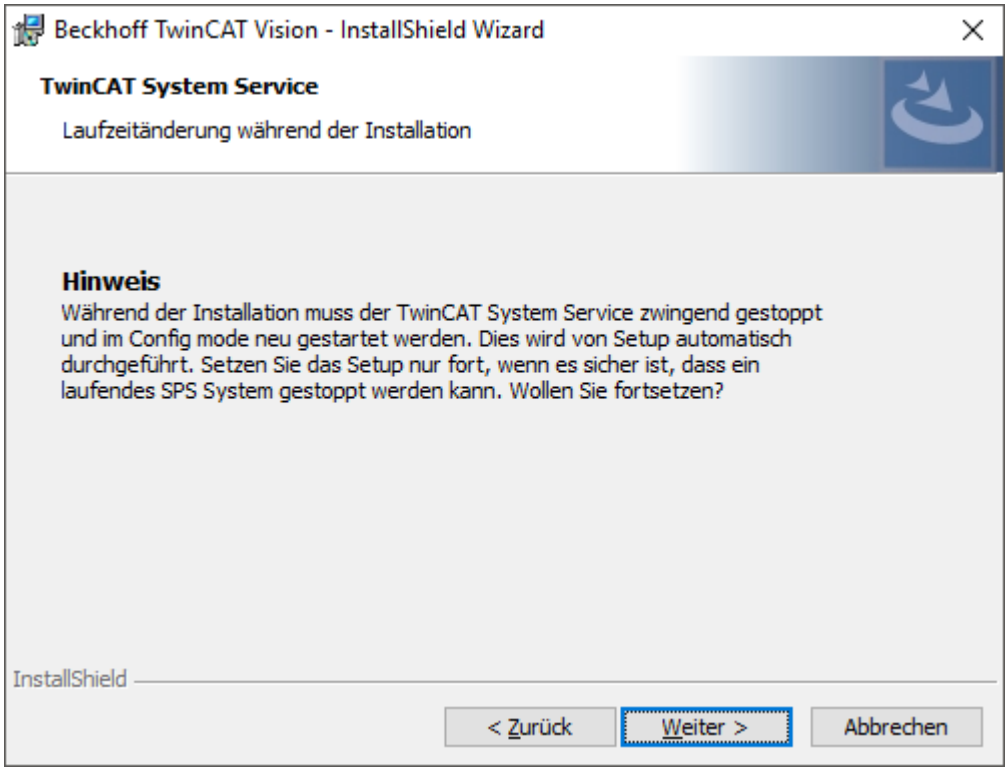


- **TC3 Vision**: Wird zum Entwickeln von TwinCAT Vision Applikationen benötigt und sollte auf XAE-Systemen installiert werden.
- **TC3 GigE Vision Connector**: Wird zum Entwickeln mit GigE Vision Geräten benötigt und sollte auf XAE-Systemen installiert werden.
- **TC3 Vision Service XAR**: Enthält den TwinCAT Vision Service und muss auf Zielsystemen einer TwinCAT Vision Applikation installiert werden.

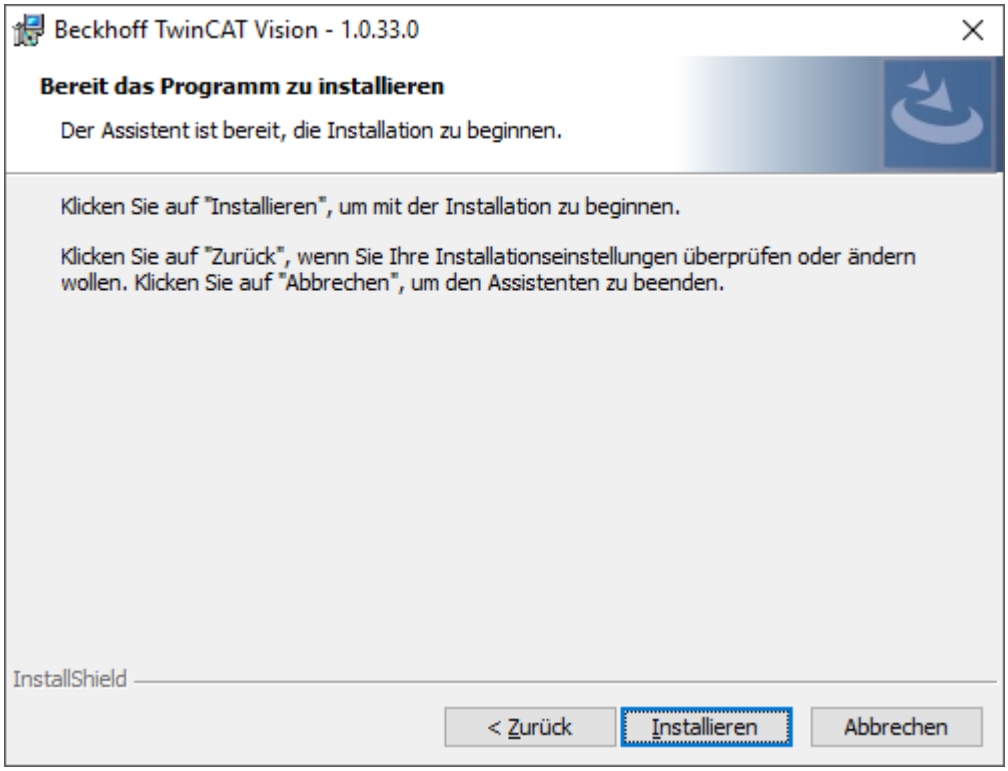
Erklärung

Der Vision Service muss auf dem Zielsystem installiert sein. Die Komponenten für TC3 Vision und TC3 GigE Vision Connector werden hingegen immer beim Aktivieren des TwinCAT Projekts automatisch von dem Entwicklungssystem auf das Zielsystem kopiert. Sie müssen daher nicht zwangsweise auf dem Zielsystem installiert werden.

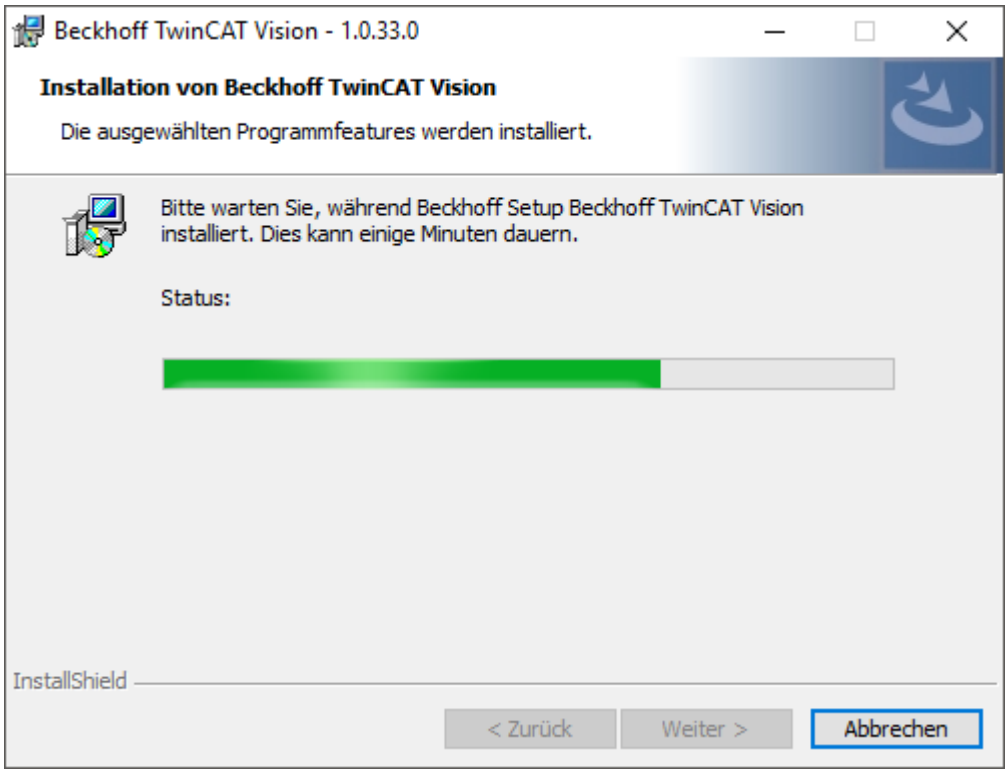
9. Nehmen Sie zur Kenntnis, dass während der folgenden Installation der TwinCAT System Service gestoppt und im Config-Mode neu gestartet wird. Fahren Sie mit der Installation nur fort, wenn ein gegebenenfalls laufendes SPS-System gestoppt werden kann. Klicken Sie in dem Fall auf **Weiter**.



10. Stellen Sie sicher, dass Sie mit den gewählten Installationseinstellungen einverstanden sind. Klicken Sie dann auf **Weiter**.

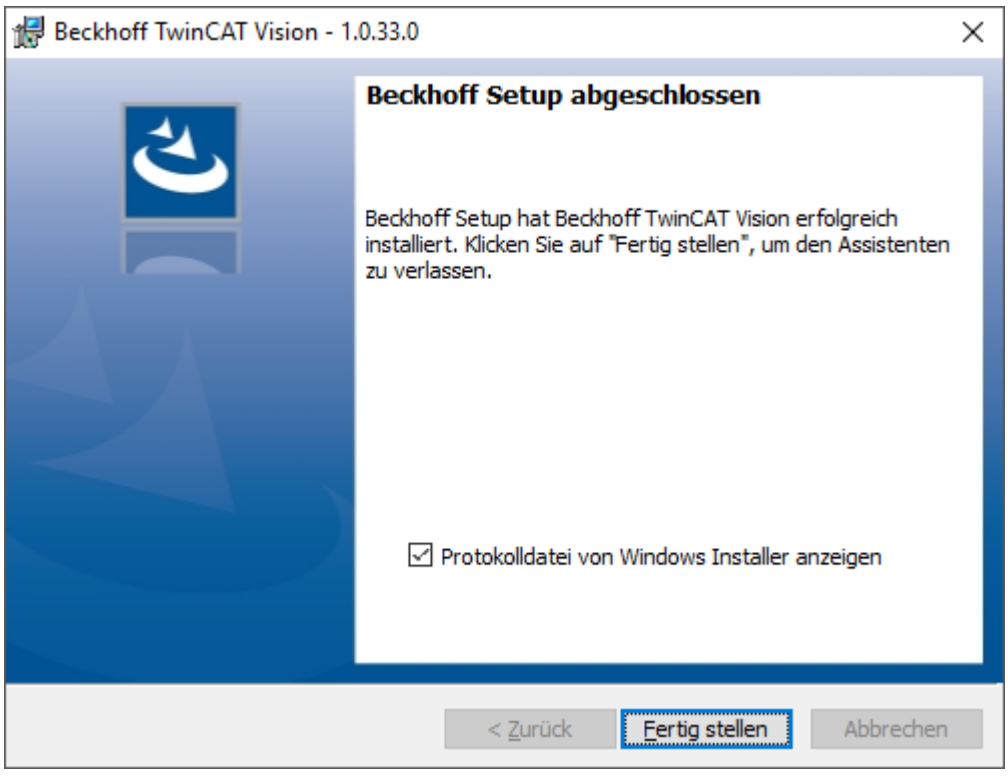


11. Warten Sie, während die gewählten TwinCAT Vision Komponenten installiert werden.



12. Ggfs. erscheint ein Dialog mit Applikationen, die für eine vollständige Installation geschlossen werden müssen. Wählen Sie aus, ob die Applikationen automatisch geschlossen werden sollen oder Sie stattdessen den PC nach der Installation neustarten möchten.

⇨ Die Installation ist nun abgeschlossen. Optional können Sie auswählen, sich die Protokolldatei der Installation anzeigen zu lassen. Klicken Sie auf **Fertig stellen**.



Bevor TwinCAT Vision nach einer erfolgreichen Installation genutzt werden kann, muss es lizenziert werden. Dies ist im Kapitel Lizenzierung [▶ 25] beschrieben.

Informationen zu den Versionen und dem Update vorhandener Projekte finden Sie in der Versionsübersicht [▶ 16].

Falls die Installation fehlerhaft ist, haben Sie folgende Optionen:

- Lösen Sie das Problem mit Hilfe der Fehlerbehebung [▶ 2948].
- Wenden Sie sich unter Bereitstellung des Installationsprotokolls an den Kundensupport. Das Installationsprotokoll *TF7xxx-Vision.exe.log* liegt in dem folgenden Ordner im Pfad: *C:\Users\<Benutzername>*

3.4 Installationsanleitung ab TwinCAT 3.1 Build 4026

TwinCAT Package Manager

Wenn Sie TwinCAT 3.1 Build 4026 (und höher) auf dem Betriebssystem Microsoft Windows verwenden, können Sie TwinCAT Vision über den TwinCAT Package Manager installieren, siehe Installation TwinCAT 3.1 Build 4026.

Normalerweise installieren Sie die TwinCAT Vision über den entsprechenden Workload; dennoch können Sie die im Workload enthaltenen Pakete auch einzeln installieren. Diese Dokumentation beschreibt im Folgenden kurz den Installationsvorgang über den Workload.

Kommandozeilenprogramm TcPkg

Über das TcPkg **C**ommand **L**ine **I**nterface (CLI) können Sie sich die verfügbaren Workloads auf dem System anzeigen lassen:

```
tcpkg list -t workload
```

Um nur die TwinCAT Vision Workloads angezeigt zu bekommen, können Sie dieses Kommando verwenden:

```
tcpkg list TF7
```

Über die folgenden Kommandos können Sie die Workloads der TF7000-TF7810 Vision-Functions installieren.

```
tcpkg install TF7xxx.Vision.XAE
tcpkg install TF7xxx.Vision.XAR
```

TwinCAT Package Manager UI

Über das **U**ser **I**nterface (UI) können Sie sich alle verfügbaren Workloads anzeigen lassen und diese bei Bedarf installieren.
Folgen Sie hierzu den entsprechenden Anweisungen in der Oberfläche.

3.5 Verwendung der TwinCAT 3 Usermode Runtime

Mit TwinCAT 3.1 Build 4026 steht eine Usermode Runtime zur Verfügung. Um TwinCAT Vision mit der TC170x | TwinCAT 3 Usermode Runtime verwenden zu können, sind die nachfolgend beschriebenen Konfigurationen zu beachten. Die grundlegende Beschreibung der Installation, Funktionsweise und Limitierungen der TwinCAT 3 Usermode Runtime sind in der zugehörigen Dokumentation zu finden.

Starten des Vision-Service

TwinCAT 3 Vision nutzt den Vision-Service für alle Interaktionen mit der Festplatte. Wenn z. B. das Laden oder Speichern von Bildern benötigt wird, was auch auf die Verwendung des FileSource zutrifft, muss der Vision-Service von der TwinCAT 3 Usermode Runtime gestartet werden.

Hierfür muss die Datei *StartManConfig.xml* mit dem folgenden Inhalt in dem *3.1\Target* Verzeichnis der Usermode Runtime angelegt oder falls vorhanden erweitert werden, Standardpfad: "*C:\ProgramData\Beckhoff\TwinCAT\3.1\Runtimes\UmRT_Default\3.1\Target*".

```
<StartMan>
    <TwinCATServers>
        <TwinCATServer Enabled="1">
            <Name>TcVnService</Name>
```

```
            <Path>C:\Program Files
(x86)\Beckhoff\TwinCAT\3.1\Components\Vision\TcVision\TcVnService.exe</Path>
        </TwinCATServer>
    </TwinCATServers>
</StartMan>
```

Verwenden einer Kamera

Um eine Kamera über den RT-Ethernet Adapter einbinden zu können, muss die Usermode Runtime mit Administratorrechten gestartet werden:

1. Starten Sie z.B. die Windows PowerShell als Administrator
2. Wechseln Sie in den Ordner der Usermode Runtime, Standardpfad „*C:\ProgramData\Beckhoff\TwinCAT\3.1\Runtimes\UmRT_Default*“ und führen die „*Start.bat*“ aus:
   `cd C:\ProgramData\Beckhoff\TwinCAT\3.1\Runtimes\UmRT_Default`
   `.\Start.bat`

⇨ Anschließend kann die Kamera über den RT-Ethernet Adapter eingebunden werden.

Da die TwinCAT 3 Usermode Runtime kein garantiert deterministisches Laufzeitverhalten hat, kann es je nach verwendeter Hardware zu Problemen bei der Kameraverbindung kommen. Beachten Sie hierzu die Diagnosemöglichkeiten der Kamerakommunikation [▶ 2964].

Bei Problemen müssen ggf. die folgenden Parameter des TcCOM-Parameter [▶ 108] Moduls erhöht werden:

- `TransmissionTimeout`
- `HeartbeatRate`
- `DiscoveryTimeout`

Beachten Sie, dass bei einer Erhöhung der Timeouts im GVCP-Modul ggf. auch das ADS Communication Timeout in Aktionen & Einstellungen [▶ 91] erhöht werden muss. Dieses Timeout sollte größer als das Produkt aus MaxTimeouts und TransmissionTimeout sein.

Erhöhung des Router-Speichers

Die Einstellung des Router-Speichers der Usermode Runtime kann in der TwinCAT Systemkonfiguration vorgenommen werden. Die einstellbare Größe ist jedoch durch den konfigurierten Heap-Speicher der Usermode Runtime begrenzt. Daher muss der Heap-Speicher größer als der einzustellende Router-Speicher gewählt werden. Der Heap-Speicher kann in der *TcRegistry.xml* im *\3.1* Verzeichnis der Usermode Runtime angepasst werden, Standardpfad: "*C:\ProgramData\Beckhoff\TwinCAT\3.1\Runtimes\UmRT_Default\3.1\TcRegistry.xml*"

Dazu muss die XML-Datei wie folgt angepasst werden:

1. Stoppen Sie die Usermode Runtime.
2. Öffnen Sie die *TcRegistry.xml.*
3. Passen Sie den Eintrag `<Value Name="HeapMemSizeMB" Type="DW">{size in MB}</Value>` in der XML-Datei an.
4. Die Größe wird in Megabyte festgelegt. In diesem Beispiel sind es 1024 MB.

```
---snipped---
<Key Name="System">
    <Value Name="RunAsDevice" Type="DW">1</Value>
    <Value Name="AmsNetId" Type="BIN">C0A804010101</Value>
    <Value Name="HeapMemSizeMB" Type="DW">1024</Value>
---snipped---
```

3.6 Lizenzierung

Die TwinCAT 3 Function ist als Vollversion oder als 7-Tage-Testversion freischaltbar. Beide Lizenztypen sind über die TwinCAT-3-Entwicklungsumgebung (XAE) aktivierbar.

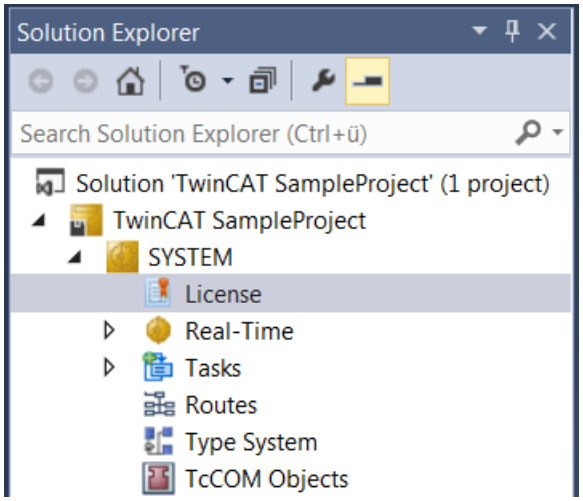
Lizenzierung der Vollversion einer TwinCAT 3 Function

Die Beschreibung der Lizenzierung einer Vollversion finden Sie im Beckhoff Information System in der Dokumentation „TwinCAT-3-Lizenzierung“.

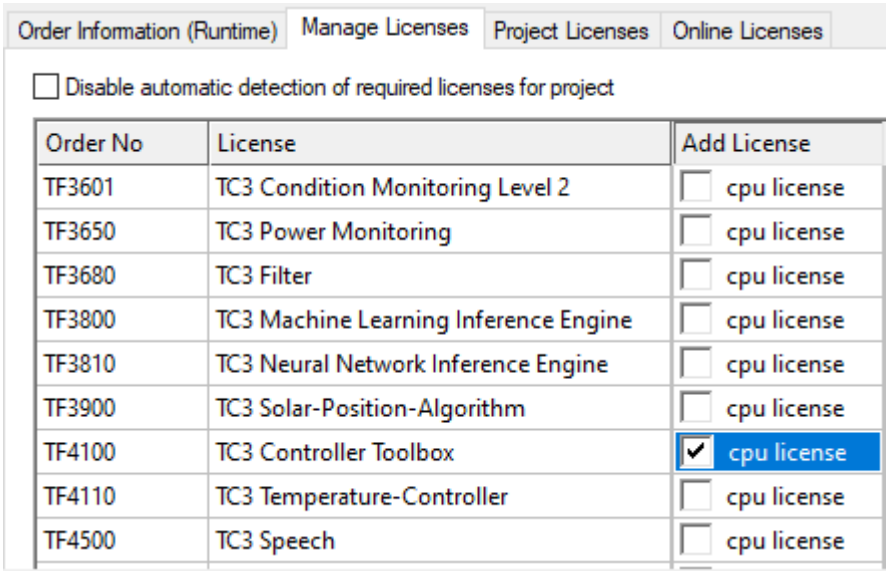
Lizenzierung der 7-Tage-Testversion einer TwinCAT 3 Function

Eine 7-Tage-Testversion kann nicht für einen TwinCAT-3-Lizenz-Dongle freigeschaltet werden.

1. Starten Sie die TwinCAT-3-Entwicklungsumgebung (XAE).
2. Öffnen Sie ein bestehendes TwinCAT-3-Projekt oder legen Sie ein neues Projekt an.
3. Wenn Sie die Lizenz für ein Remote-Gerät aktivieren wollen, stellen Sie das gewünschte Zielsystem ein. Wählen Sie dazu in der Symbolleiste in der Drop-down-Liste **Choose Target System** das Zielsystem aus.
   ⇨ Die Lizenzierungseinstellungen beziehen sich immer auf das eingestellte Zielsystem. Mit der Aktivierung des Projekts auf dem Zielsystem werden automatisch auch die zugehörigen TwinCAT-3-Lizenzen auf dieses System kopiert.
4. Klicken Sie im **Solution Explorer** im Teilbaum **SYSTEM** doppelt auf **License.**

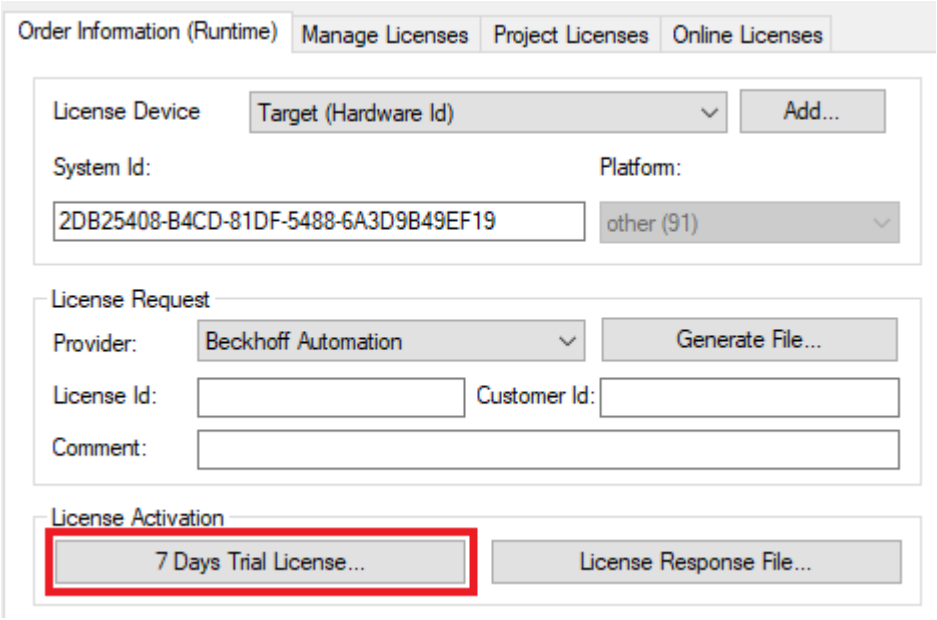


   ⇨ Der TwinCAT-3-Lizenzmanager öffnet sich.
5. Öffnen Sie die Registerkarte **Manage Licenses.** Aktivieren Sie in der Spalte **Add License** das Auswahlkästchen für die Lizenz, die Sie Ihrem Projekt hinzufügen möchten (z. B. „TF4100 TC3 Controller Toolbox“).

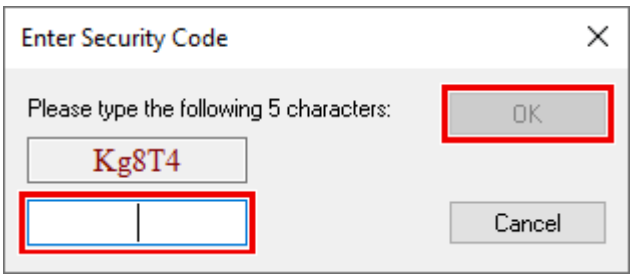


Order No	License	Add License
TF3601	TC3 Condition Monitoring Level 2	cpu license
TF3650	TC3 Power Monitoring	cpu license
TF3680	TC3 Filter	cpu license
TF3800	TC3 Machine Learning Inference Engine	cpu license
TF3810	TC3 Neural Network Inference Engine	cpu license
TF3900	TC3 Solar-Position-Algorithm	cpu license
TF4100	TC3 Controller Toolbox	cpu license
TF4110	TC3 Temperature-Controller	cpu license
TF4500	TC3 Speech	cpu license

6. Öffnen Sie die Registerkarte **Order Information (Runtime).**
   ⇨ In der tabellarischen Übersicht der Lizenzen wird die zuvor ausgewählte Lizenz mit dem Status „missing“ angezeigt**.**
7. Klicken Sie auf **7 Days Trial License...**, um die 7-Tage-Testlizenz zu aktivieren.



   ⇨ Es öffnet sich ein Dialog, der Sie auffordert, den im Dialog angezeigten Sicherheitscode einzugeben.



8. Geben Sie den Code genauso ein, wie er angezeigt wird, und bestätigen Sie ihn.
9. Bestätigen Sie den nachfolgenden Dialog, der Sie auf die erfolgreiche Aktivierung hinweist.
   ⇨ In der tabellarischen Übersicht der Lizenzen gibt der Lizenzstatus nun das Ablaufdatum der Lizenz an.
10. Starten Sie das TwinCAT-System neu.

⇨ Die 7-Tage-Testversion ist freigeschaltet.

4 First Steps

Der folgende Leitfaden bietet eine Einführung in die Verwendung von TwinCAT Vision (nach erfolgreicher Installation [▶ 15]).

1. Projekt erstellen [▶ 28]
   - TwinCAT Projekt erstellen [▶ 28]
   - Zielsystem auswählen [▶ 30]
   - System konfigurieren [▶ 32]
2. Vision-Gerät einrichten [▶ 34] anzeigen und Vision-Gerät anlegen
   - Vision-Knoten einblenden und Applikation erstellen [▶ 34]
   - File Source Control anlegen [▶ 36] oder
   - GigE Vision Kamera anlegen [▶ 39]
3. SPS-Projekt erstellen [▶ 47]
   - SPS-Projekt erstellen [▶ 47]
   - SPS-Bibliothek einbinden [▶ 48]
   - SPS-Programm schreiben [▶ 50]
   - Funktionsbaustein initialisieren [▶ 51]
4. Ausführen [▶ 52]
   - TwinCAT-Projekt aktivieren und SPS-Projekt starten [▶ 52]
   - Bildanzeige im ADS Image Watch [▶ 53]
   - Download Beispielprojekt: https://github.com/Beckhoff/TF7xxx_Samples

Detailliertere Informationen sind unter Entwicklungsumgebung [▶ 55], API-Referenz [▶ 146] und Beispiele [▶ 2838] zu finden.

4.1 Projekt erstellen

Video-Beispiel zum Einrichten eines Systems für TwinCAT Vision:

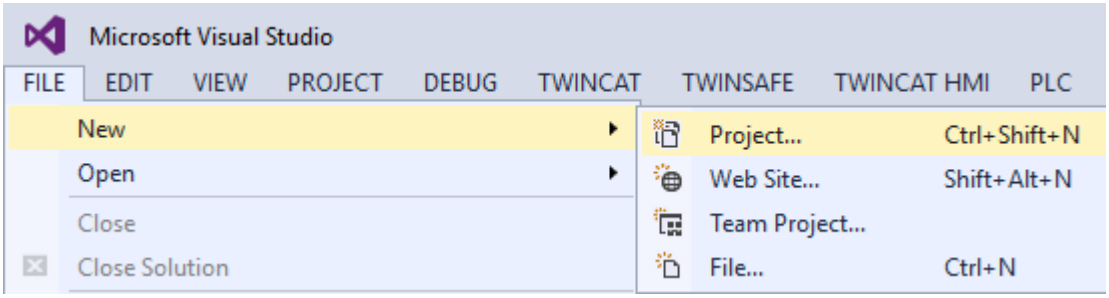
Video:: https://infosys.beckhoff.com/content/1031/tf7xxx_tc3_vision/Resources/5906036107.mp4

4.1.1 TwinCAT-Projekt erstellen

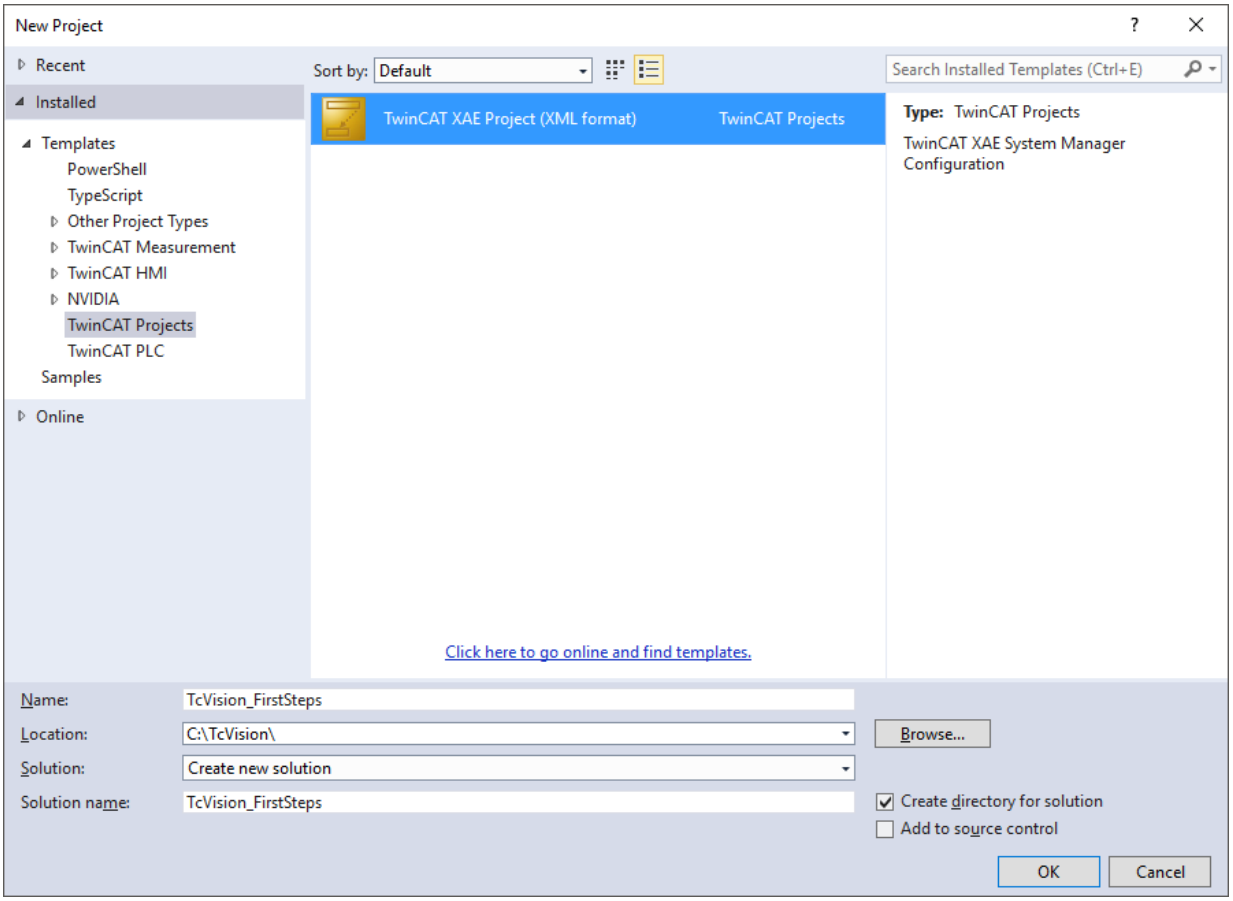
TwinCAT Vision ist vollständig in das TwinCAT-System integriert. Ein Standard TwinCAT-Projekt ist daher Ausgangspunkt für jede TwinCAT Vision Applikation:

1. Öffnen Sie Visual Studio mit integriertem TwinCAT.

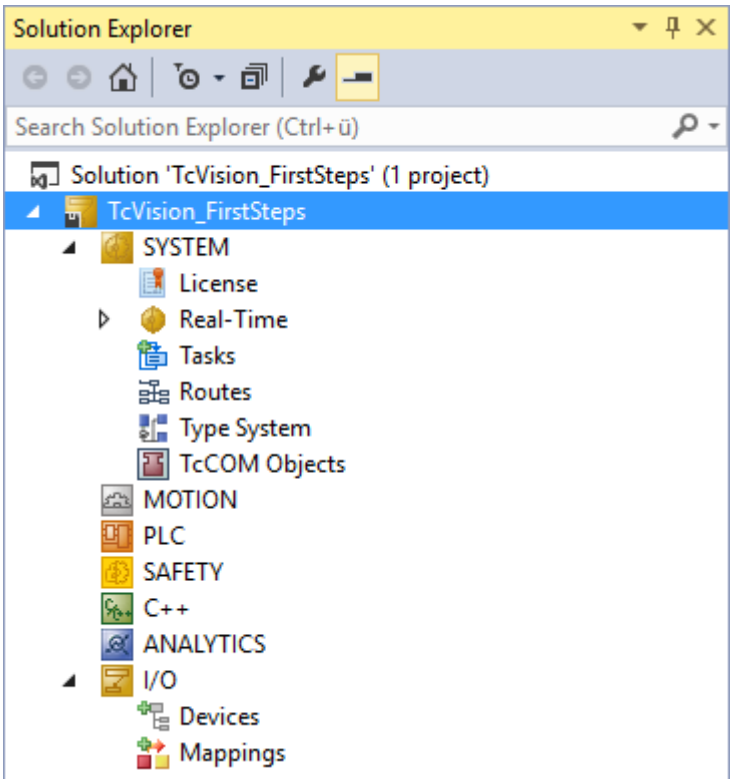
2. Öffnen Sie den Dialog **Neu/Projekt**.



3. Wählen Sie Namen und Speicherort.



⇨ Ein TwinCAT-Projekt wird angelegt.



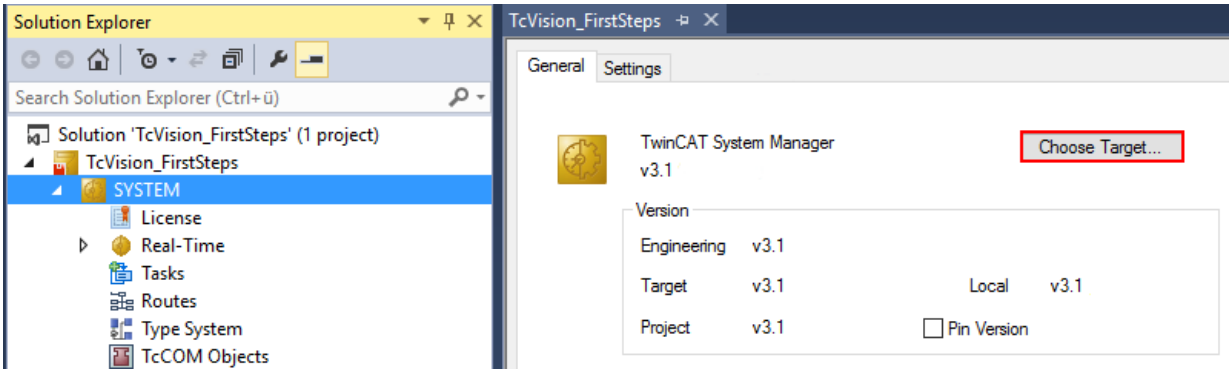
Nächster Schritt

Zielsystem auswählen [▶ 30]

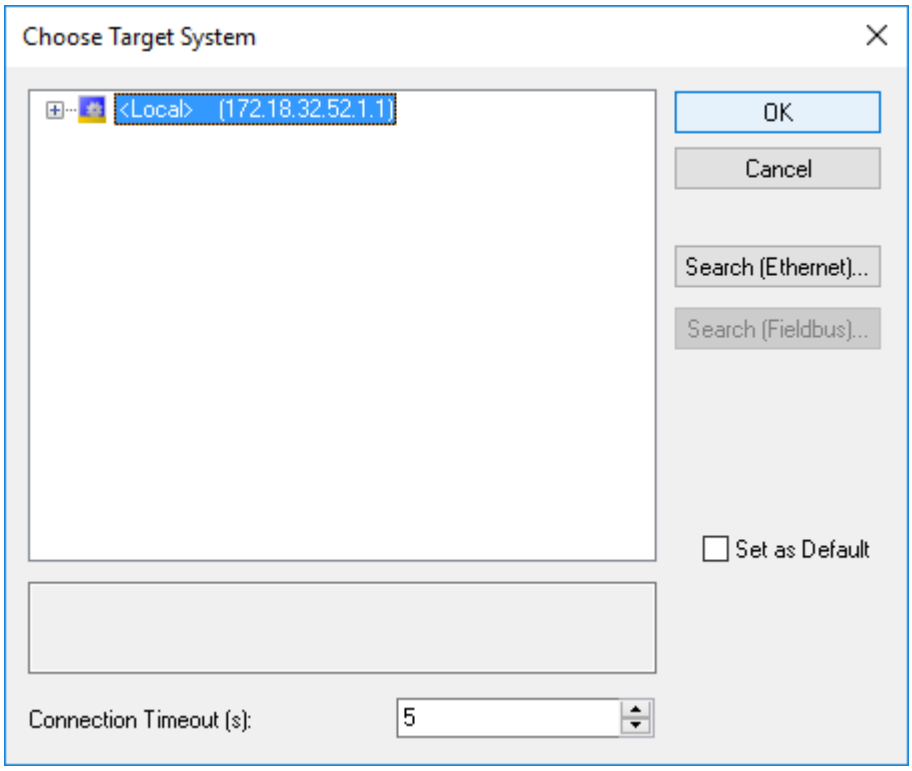
4.1.2 Zielsystem auswählen

In TwinCAT wird zwischen Entwicklungs- und Laufzeitsystem unterschieden. Sofern Ihr Entwicklungssystem nicht ebenfalls Ihr Laufzeitsystem ist, erstellen Sie ggfs. eine Route zum Laufzeitsystem und wählen Sie diese aus. Dies ist unter anderem über die nachfolgenden Schritte möglich:

1. Klicken Sie auf **Solution Explorer > SYSTEM > General > Choose Target...**
   Es öffnet sich der **Choose Target System**-Dialog, der alle Zielsysteme aufführt, zu denen Routen vom Entwicklungssystem bestehen.

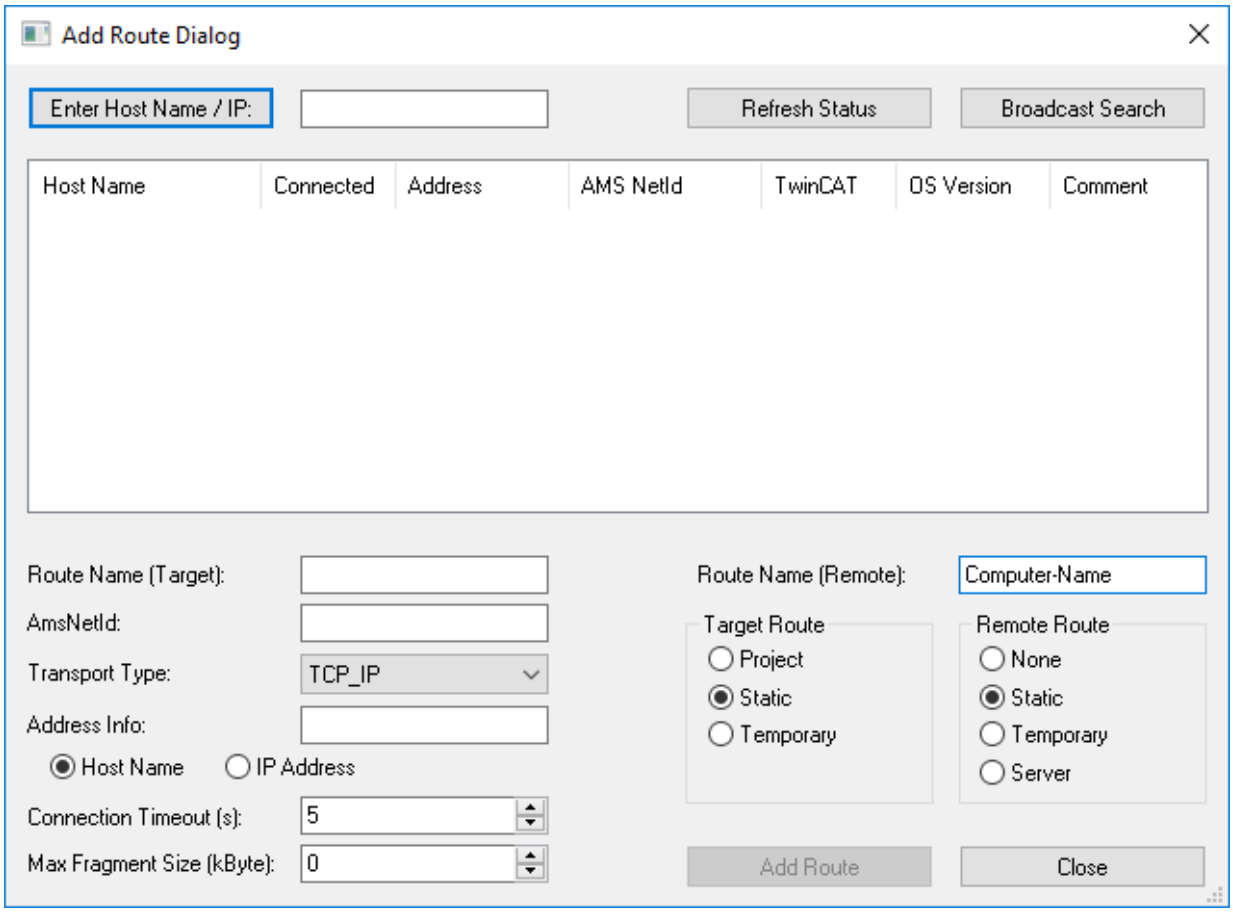


2. Ist das Zielsystem noch nicht aufgeführt, das heißt es besteht noch keine Route zum Systemen, klicken Sie auf **Search**.

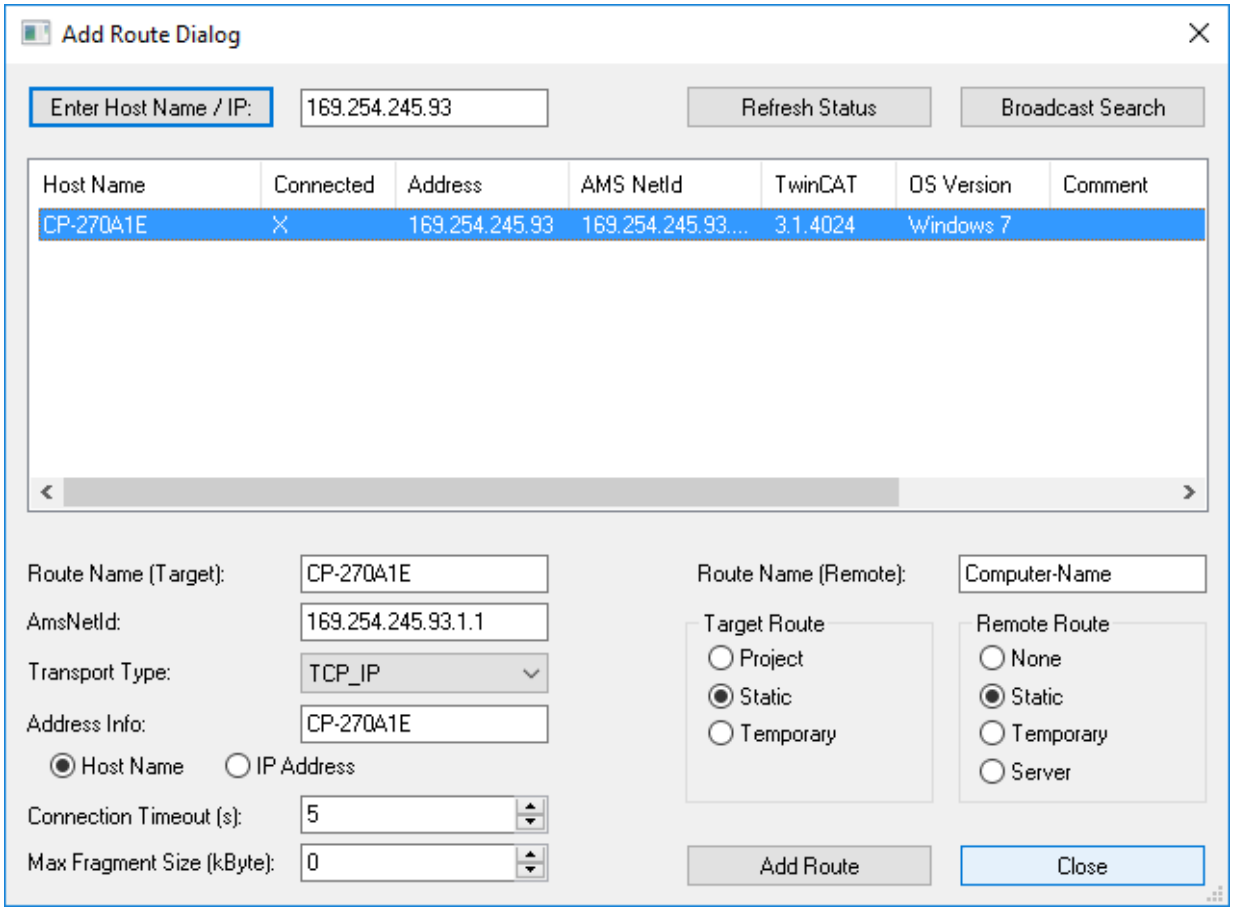


Der Add Route Dialog öffnet sich und Sie können weitere Routen hinzufügen.

3. Im **Add Route Dialog** klicken Sie auf **Broadcast Search** um nach TwinCAT Systemen im Netzwerk zu suchen. Alternativ suchen Sie nach dem Gerätenamen oder der IP-Adresse über **Enter Host Name / IP**. Die gefundenen Systeme werden anschließend angezeigt.



4. Ist ein System in der Liste ausgewählt, klicken Sie auf **Add Route**, um eine Route zu diesem zu erstellen. Im Kapitel Add Route Dialog sind im Detail die angezeigten Informationen und Einstellmöglichkeiten beschrieben. Die neue Route erscheint nun im **Choose Target System** Dialog.



5. Wählen Sie das Zielsystem im **Choose Target System** Dialog aus und bestätigen Sie über **OK**.

Nächster Schritt

System konfigurieren [▶ 32]

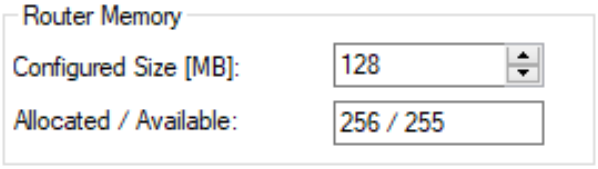
4.1.3 System konfigurieren

Die Systemkonfiguration ist abhängig vom Zielsystem und ist im Detail in Kapitel Systemkonfiguration [▶ 56] beschrieben.

Router-Speicher vergrößern

In TwinCAT Vision ist unter anderem beim Erstellen eines Bildes ein dynamisches Speicherallokieren erforderlich, welches im Router-Speicher [▶ 56] erfolgt. Dementsprechend muss der Router-Speicher groß genug konfiguriert werden.

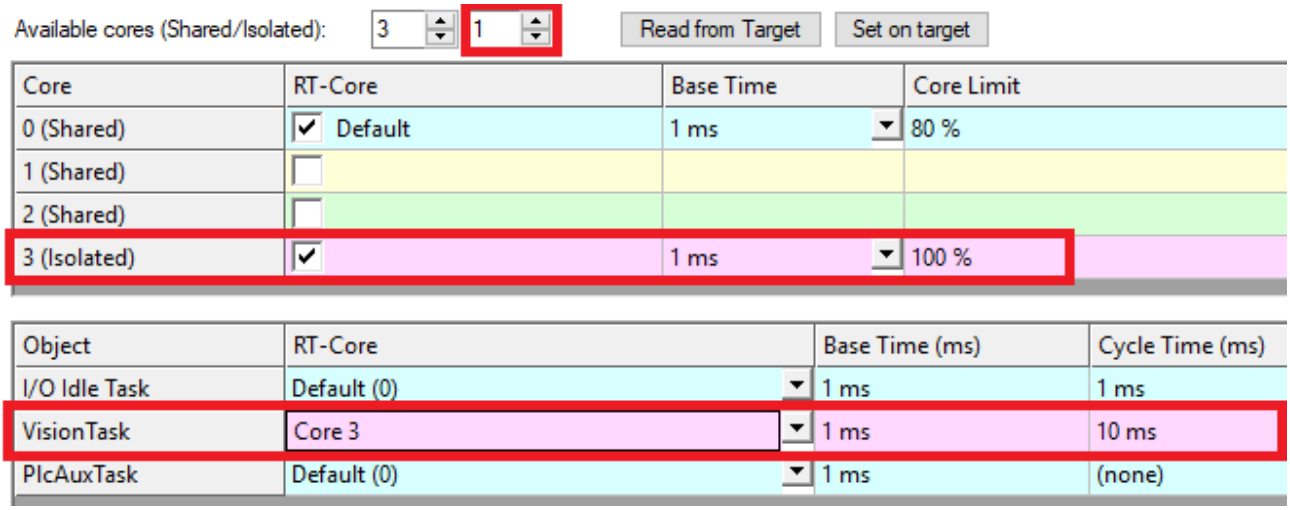
Dies erfolgt im TwinCAT-Projekt unter **SYSTEM > Real-Time > Settings > Router Memory (Mbyte)**.



Anschließend muss die Konfiguration aktiviert werden . Bei einer Verkleinerung des Router-Speichers ist zudem ein Neustart des Zielsystems erforderlich.

Auswahl eines CPU-Kerns und einer Task

Bei Multicore-Systemen ermöglicht TwinCAT die Unterscheidung zwischen Windows-Kernen und isolierten Kernen. Bei Windows-Kernen teilen sich das Betriebssystem und die TwinCAT-Anwendung die Prozessorzeit. Der Echtzeitanteil kann dabei auf 10% bis 90% begrenzt werden. Im Gegenzug stehen die isolierten Kerne vollständig der TwinCAT-Anwendung zur Verfügung. Aus diesem Grund wird die Verwendung von isolierten Kernen für Vision Anwendungen empfohlen. Zum ersten Austesten können ohne Probleme auch geteilte Kerne verwendet werden.



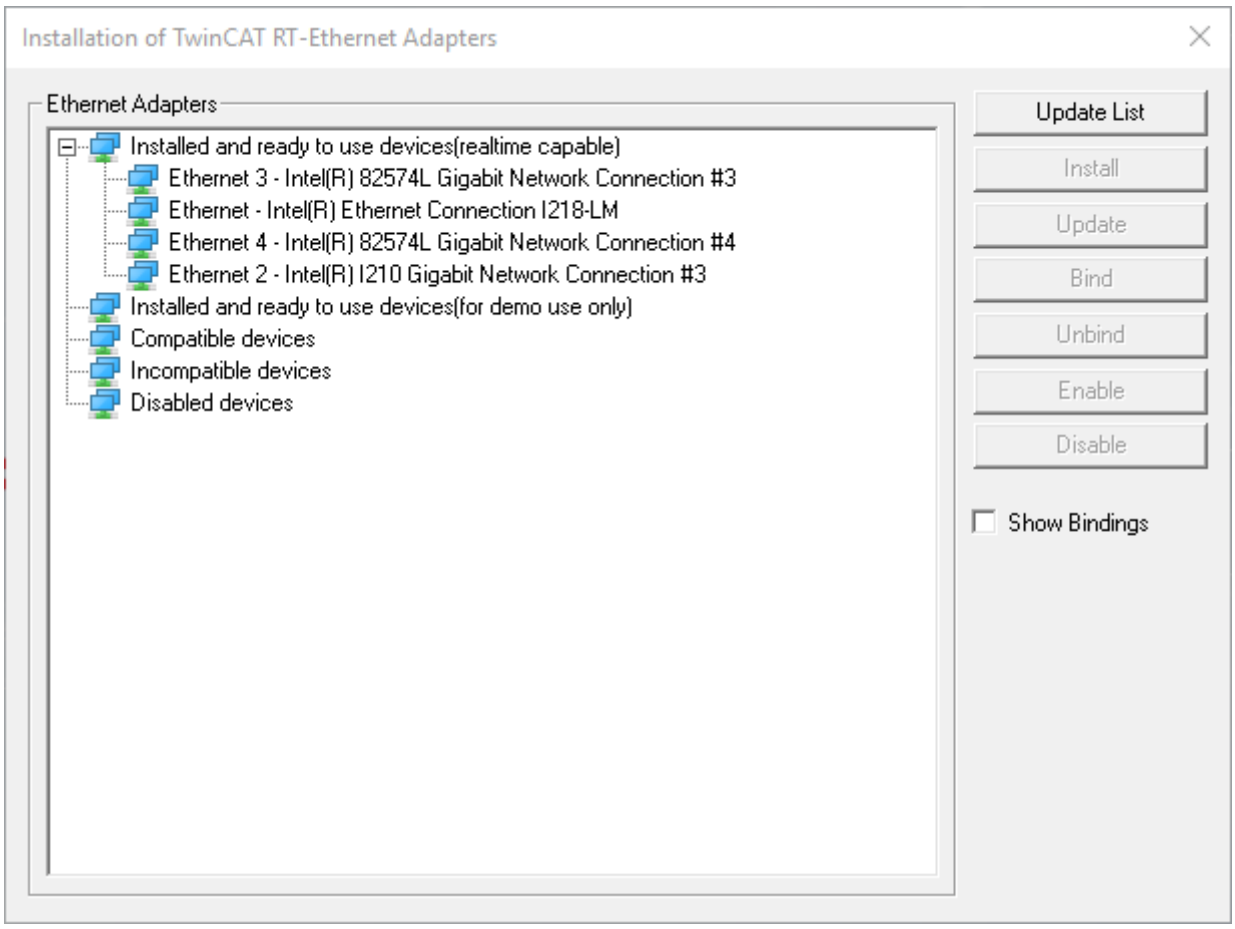
Die Kernkonfiguration erfolgt im TwinCAT Projektbaum unter **SYSTEM > Real-Time > Settings**. Ein detaillierte Anleitung zu der Konfiguration finden Sie im Kapitel CPU-Kerne und Tasks [▶ 63]. Dort ist ebenfalls beschrieben, Sie sie Zykluszeit der ausführenden Task verändern können.

Anschließend muss die Konfiguration aktiviert werden . Bei einer neuen Festlegung von geteilten und isolierten Kernen ist zudem ein Neustart des Zielsystems erforderlich.

Installation eines Treibers für Echtzeit-Ethernet

Um GigE Vision Kameras an TwinCAT Vision anzuschließen, muss auf den benutzen Netzwerkadaptern der TwinCAT RT-Ethernet Treiber installiert sein. Der Treiber sorgt dafür, dass TwinCAT den Netzwerkadapter während der Laufzeit benutzen kann. Details dazu finden Sie im Kapitel Netzwerkadapter [▶ 58].

In der Entwicklungsumgebung kann ein entsprechender Installationsassistent über **TWINCAT > Show Realtime Ethernet Compatible Devices**... aufgerufen werden. Auf dem Zielsystem kann der Assistent auch über die Datei *C:\TwinCAT\3.1\System\TcRteInstall.exe* gestartet werden. Im nun geöffneten Dialog kann für die unter **Compatible devices** gelisteten Netzwerkadapter der RT-Ethernet Treiber installiert werden.



Stellen Sie sicher, dass nun alle Netzwerkadapter, über die Sie eine Kamera verbinden möchten, in der Liste **Installed and ready to use devices (realtime capable)** stehen.

Nächster Schritt

Vision-Knoten einblenden und Applikation erstellen [▶ 34]

4.2 Vision-Gerät einrichten

Video-Beispiel zur Arbeit mit dem File Source Control:

Video:: https://infosys.beckhoff.com/content/1031/tf7xxx_tc3_vision/Resources/5931182859.mp4

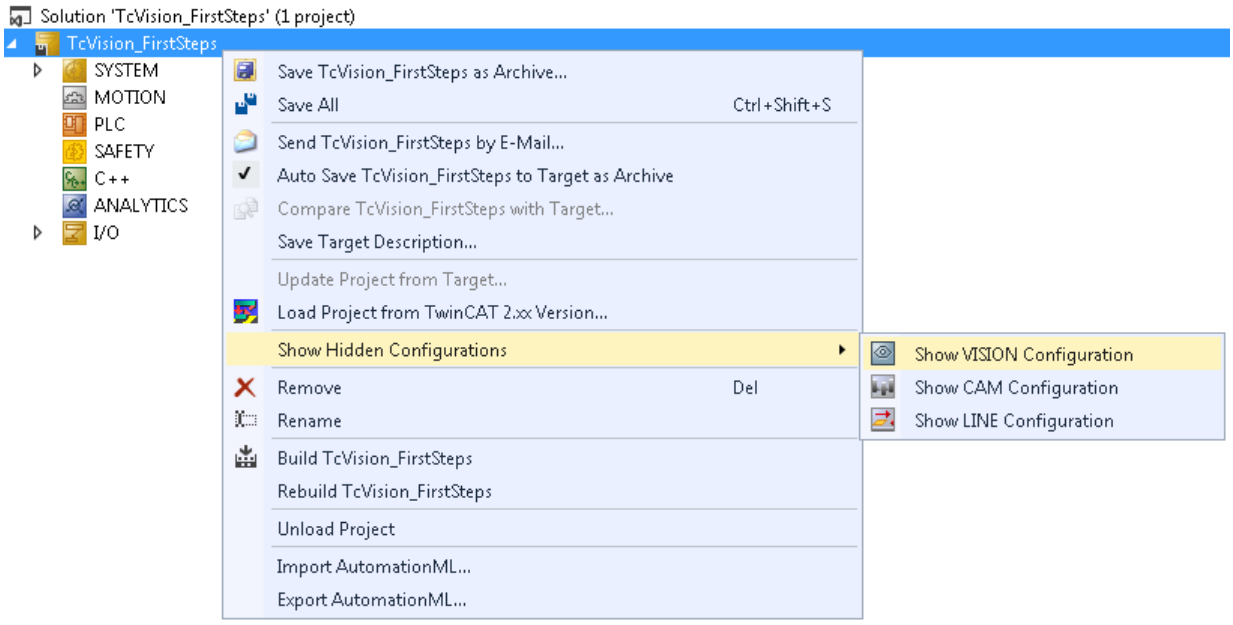
Video-Beispiel zum Anlegen einer Kamera Instanz:

Video:: https://infosys.beckhoff.com/content/1031/tf7xxx_tc3_vision/Resources/5931185163.mp4

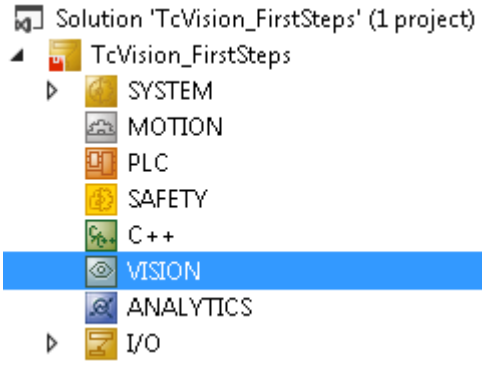
4.2.1 Vision-Knoten einblenden und Applikation erstellen

Für die Vision-Konfiguration gibt es innerhalb des TwinCAT-Projektes den Vision-Knoten.

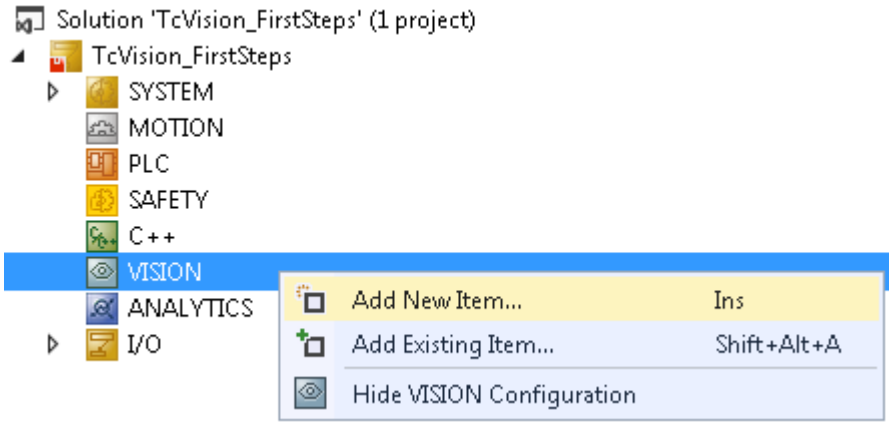
1. Falls der Vision-Knoten im TwinCAT-Projekt nicht angezeigt wird, machen Sie einen Rechtsklick auf das TwinCAT-Projekt im Solution Explorer >**Show Hidden Configurations** > **Show VISION Configuration.**



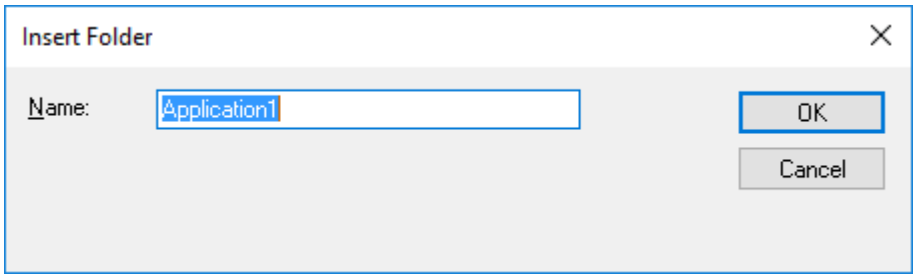
⇨ Anschließend ist der Vision-Knoten sichtbar.



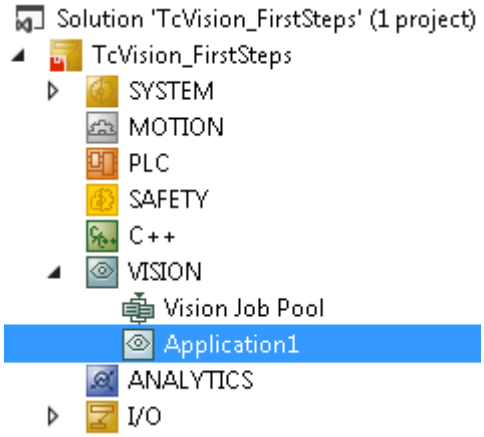
2. Innerhalb des Vision-Knotens werden Ordner - auch Applikationen genannt – verwendet, um die Bildverarbeitungsgeräte zu gruppieren.
Machen Sie einen Rechtsklick auf die **Vision-Node** > **Add New Item.**



3. Wählen Sie einen Namen aus und bestätigen Sie mit **OK.**



⇨ Der Ordner ist erstellt



Außerdem finden Sie hier weiterführende Informationen:

- VISION Knoten [▶ 68]
- Application Knoten [▶ 71]

Nächster Schritt

Im nächsten Schritt wird ein Vision-Gerät angelegt, mit dem Bilder in die SPS geladen werden können. Dies kann entweder eine reale GigE Vision Kamera oder ein sog. File Source Control zum Laden von bestehenden Bilddateien sein:

GigE Vision Kamera anlegen [▶ 39] oder File Source Control anlegen [▶ 36]

4.2.2 File Source Control anlegen

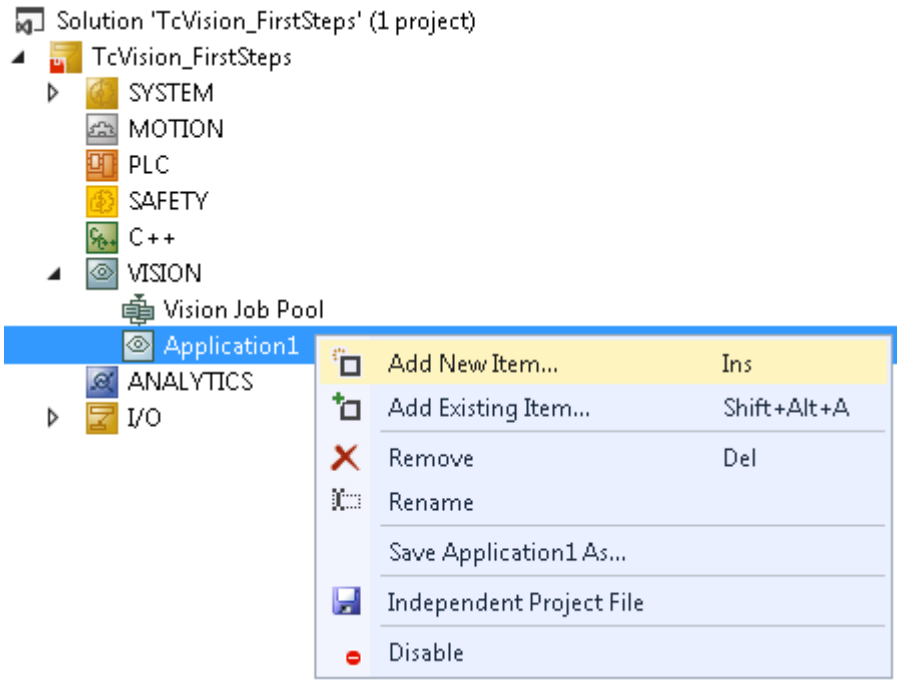
In den nachfolgenden Schritten wird das Anlegen eines File Source [▶ 135] Objekts beschrieben. Mit diesem ist es möglich, gespeicherte Bilder in die SPS zu laden. Hierzu muss sich das Zielsystem im Config Mode befinden. Klicken Sie dazu auf das Symbol in der TwinCAT XAE Base Toolbar oder im **Visual Studio Menü > TWINCAT >** auf **Restart TwinCAT (Config Mode**).

Alternativ können Sie auch eine GigE Vision Kamera anlegen [▶ 39].

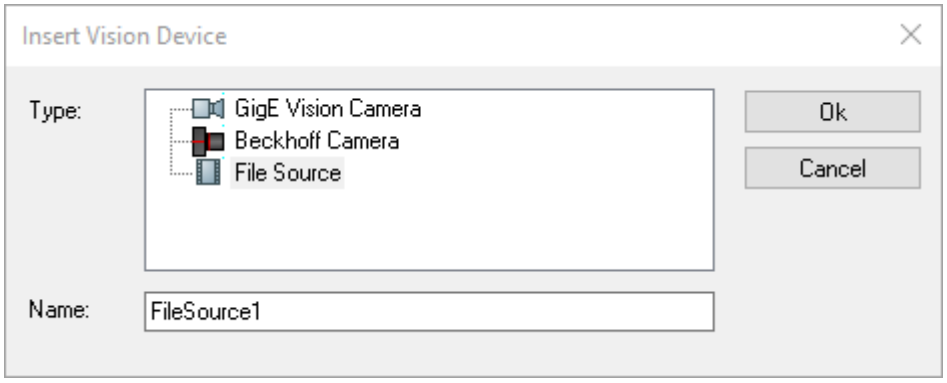
Zum Anlegen des File Source Objekts gehen Sie folgendermaßen vor:

1. Klicken Sie auf eine Applikation unterhalb des VISION-Knotens, die Sie zuvor erstellt haben.

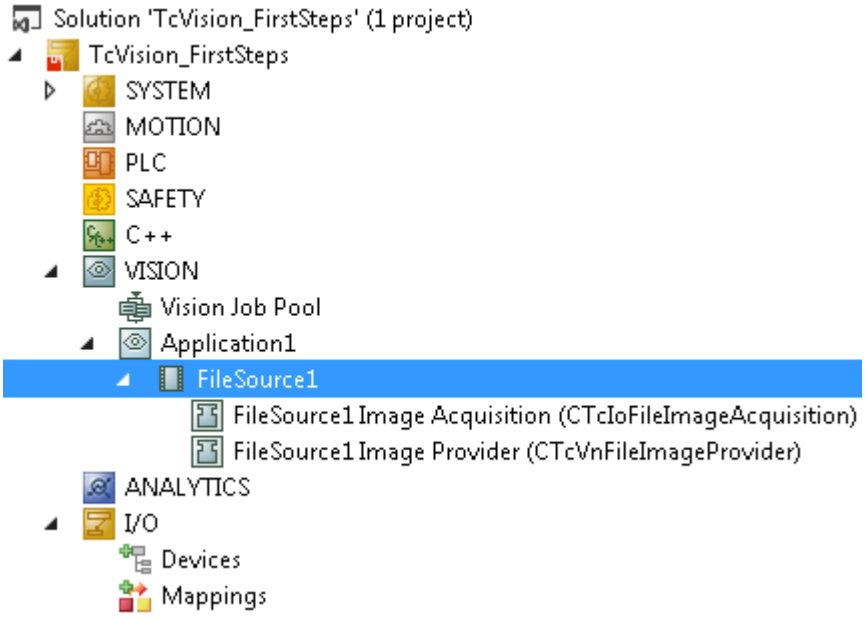
2. Nach einem Rechtsklick wählen Sie **Add new item...**



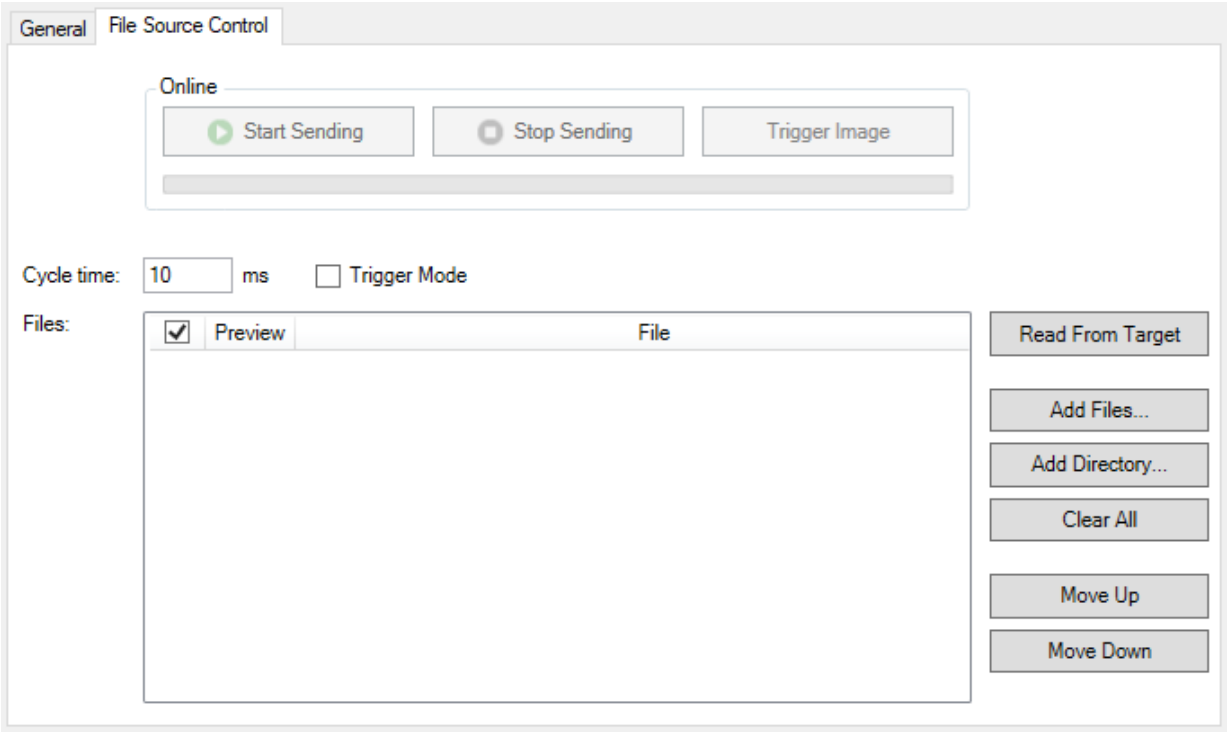
3. Wählen Sie den Geräte-Type **File Source** aus, bei Bedarf passen Sie den Namen an, klicken Sie auf **Ok**.



⇨ Das File Source Control ist nun unterhalb der Applikation angelegt und kann verwendet werden.

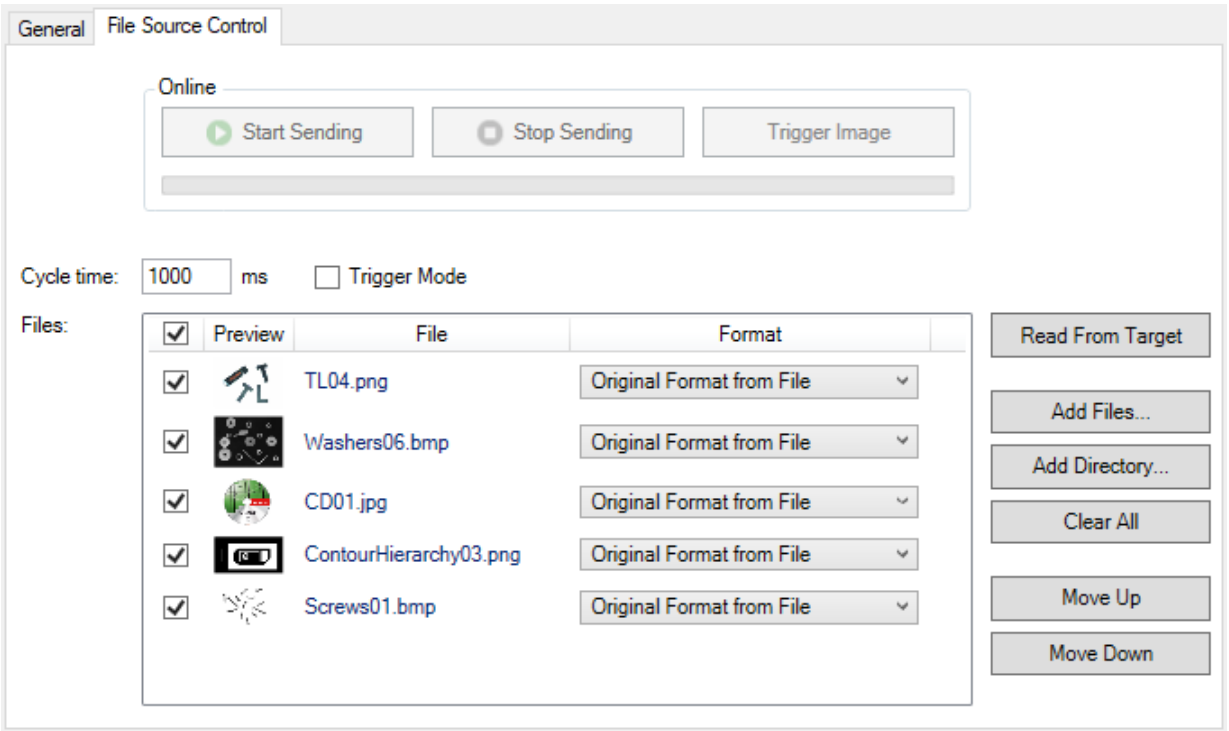


4. Klicken Sie doppelt auf das **File Source Objekt** unterhalb des Applikation-Knotens im Projektbaum. Wählen Sie im sich öffnenden Fenster den Karteireiter **File Source Control** aus und nehmen Sie hier die wesentlichen Einstellungen vor.



5. Klicken Sie auf **Read From Target**. Dabei erfolgt ein Abgleich des File Source Controls [▶ 136] im Projekt mit den evtl. hinterlegten Bildern auf dem ausgewählten Zielsystem.
6. Fügen Sie über **Add Files...**, **Add Directory...** oder per Drag and Drop Bilder zum File Source Control hinzu. Bei Bedarf können Sie sich Beispielbilder mit dem Projekt-Download herunterladen: First Steps [▶ 28]
7. Passen Sie bei Bedarf die Zykluszeit an, in der das File Source Objekt Bilder an die SPS sendet.

⇨ So kann das File Source Objekt verwendet werden:



Nächster Schritt

SPS-Projekt erstellen [▶ 47]

4.2.3 GigE Vision Kamera anlegen

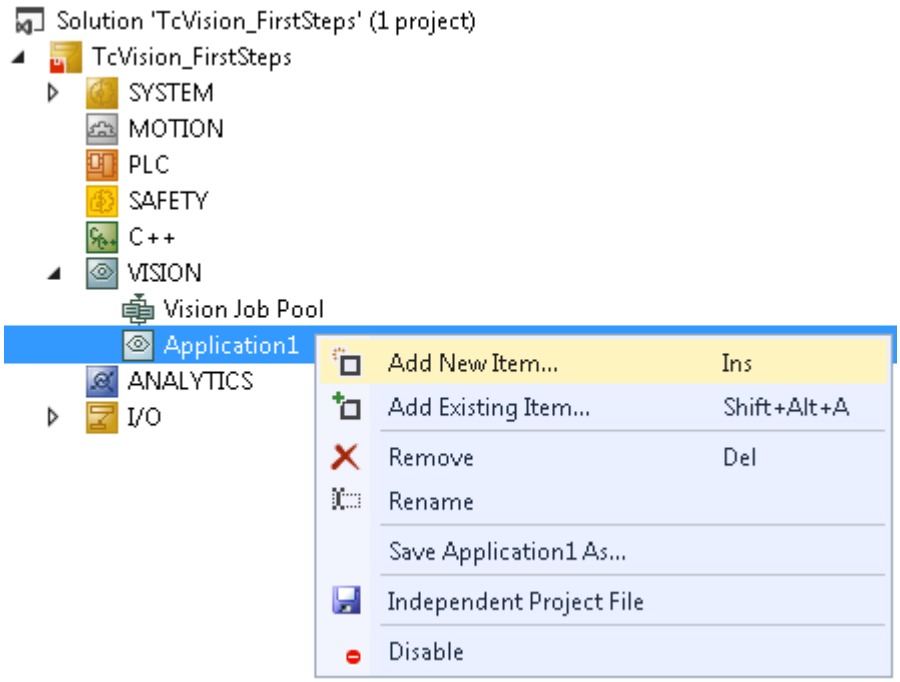
In den nachfolgenden Schritten wird das Anlegen eines GigE Vision Kamera [▶ 72]-Objekts beschrieben, die genauso zum Anlegen eines Beckhoff-Kamera [▶ 135]-Objekts verwendet werden können. Dies setzt voraus, dass eine GigE Vision Kamera am PC angeschlossen ist und für den entsprechenden Netzwerk-Port, wie in System konfigurieren [▶ 32] beschrieben, der TwinCAT RT-Ethernet Adapter installiert ist.

Zudem muss sich das Zielsystem im Config Mode befinden. Klicken Sie dazu auf das Symbol in der TwinCAT XAE Base Toolbar oder im **Visual Studio Menü > TWINCAT >** auf **Restart TwinCAT (Config Mode**).

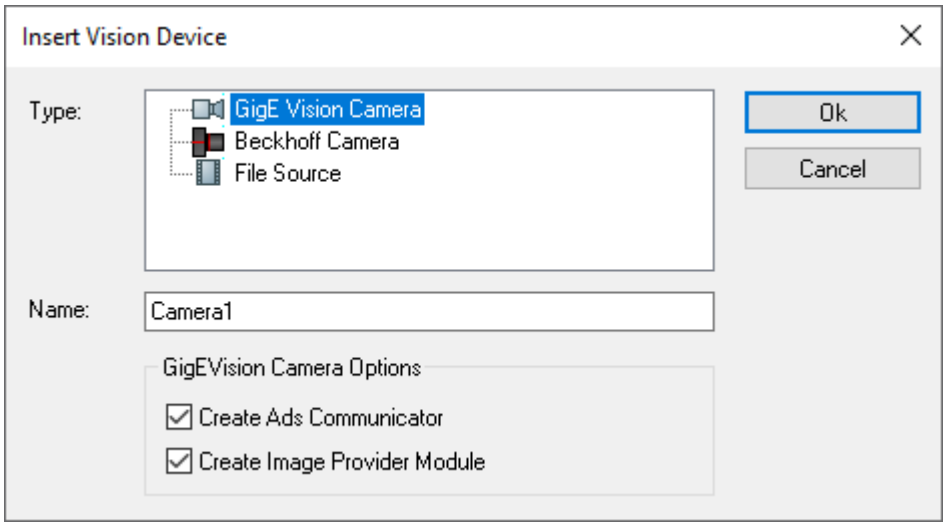
Alternativ können Sie ein File Source Control anlegen [▶ 36].

Zum Kamera Anlegen gehen Sie folgendermaßen vor:

1. Klicken Sie auf eine Applikation unterhalb des Vision-Knotens, die Sie zuvor erstellt haben.
2. Nach einem Rechtsklick wählen Sie **Add new item...**



3. Wählen Sie den Geräte-Typ **GigE Vision Camera oder Beckhoff-Kamera**. Bei Bedarf passen Sie den Kameranamen und die GigEVision Camera Options an und klicken Sie auf **Ok**.

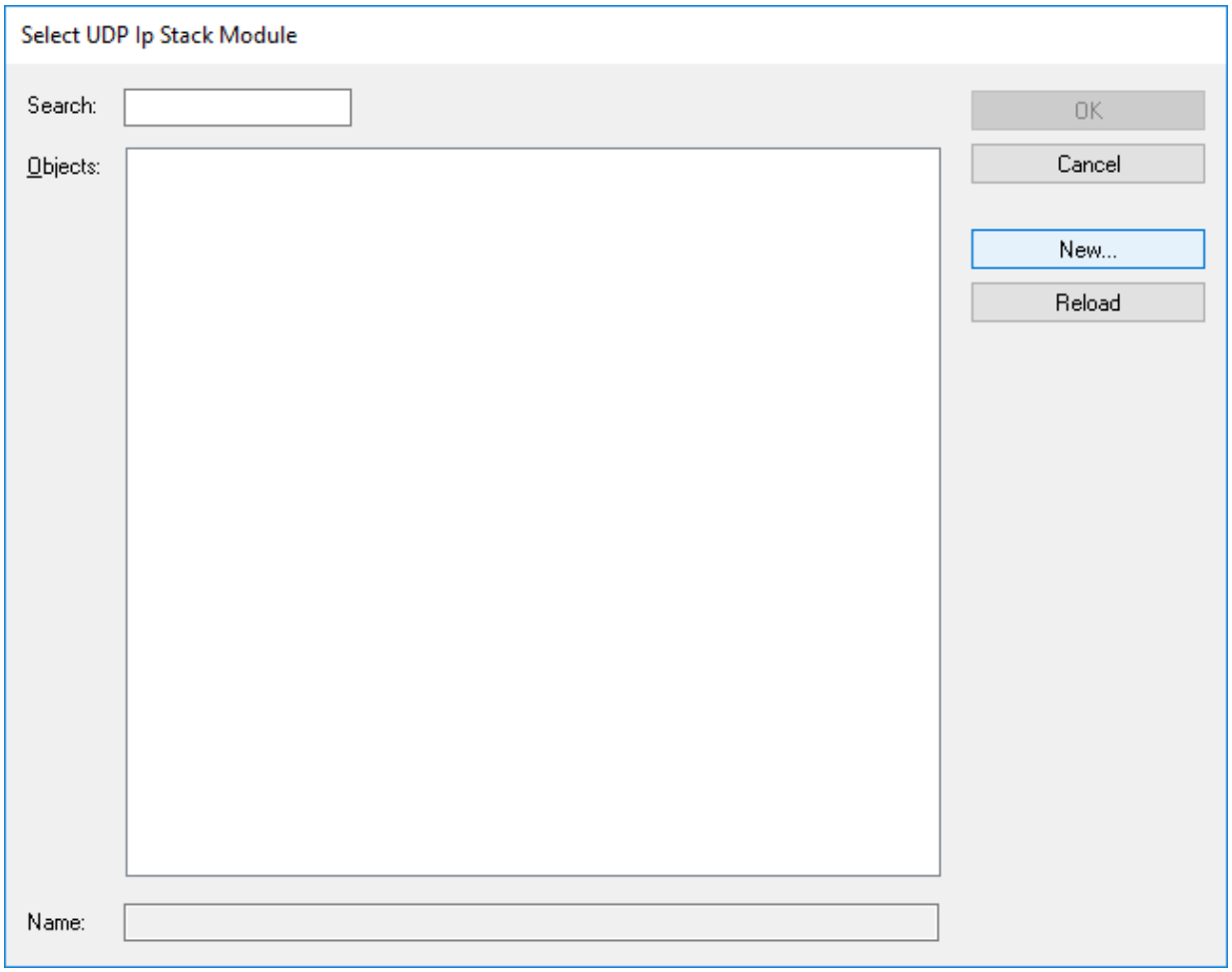


Ein Image Provider ist die Schnittstelle zu einsprechenden PLC-Funktionsblöcken und wird daher im Normalfall immer benötigt (siehe Kamera-Details [▶ 132]).

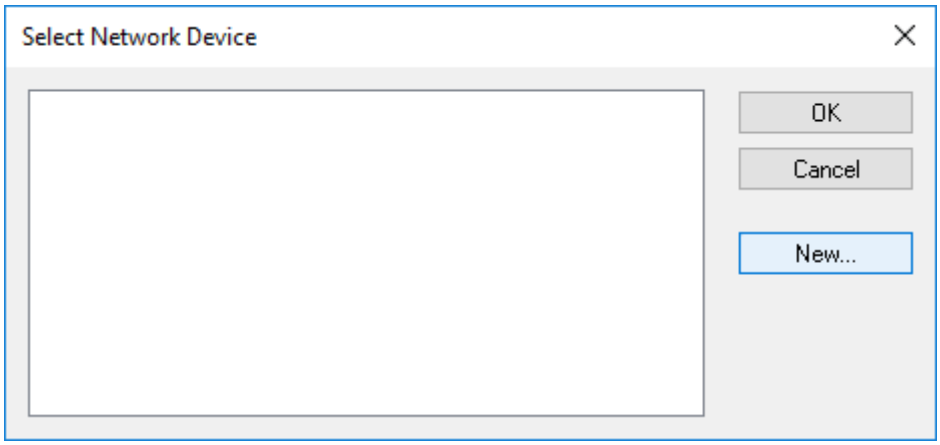
Ein Ads Communicator wird nur zum Aufzeichnen von Streams über den Record/Playback [▶ 109] Karteireiter im Kameraobjekt benötigt.
Wenn keine Streams aufgezeichnet werden sollen, kann diese Option abgewählt werden. Dadurch entfällt das Ads Communicator Module sowie die entsprechende VnTask ADS.

4. GigE Vision Kameras kommunizieren über UDP/IP. Dafür wird für den verwendeten Netzwerk Port ein entsprechendes Netzwerk-Gerät mit entsprechendem IP-Stack im IO-Knoten benötigt. Legen Sie dies mit einem Klick auf **New...** an.
Für jeden physikalischen Port einer Netzwerkkarte darf nur ein Netzwerk-Gerät mit nur einem IP-

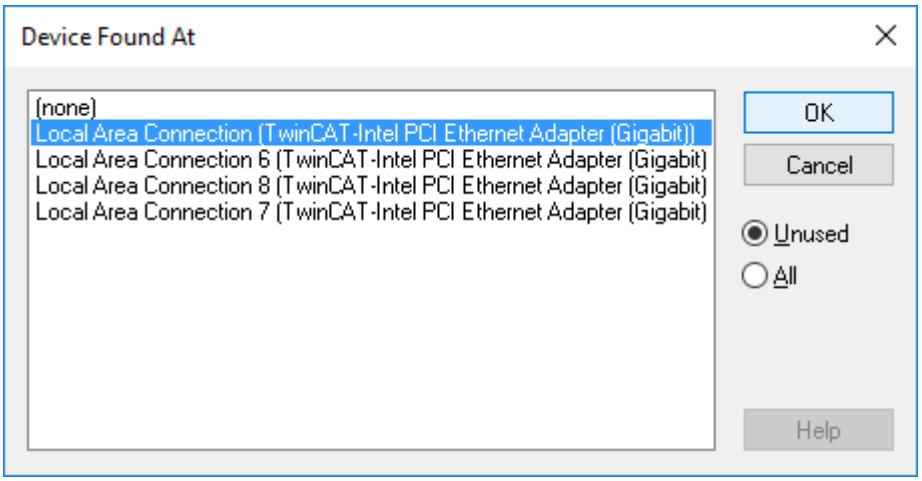
Stack angelegt werden.



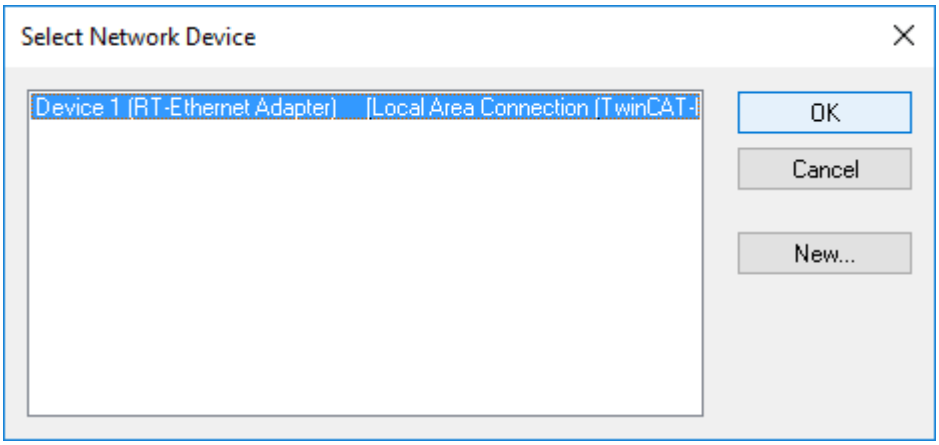
5. Um ein neues Netzwerk-Gerät anzulegen, klicken Sie im **Select Network Device**-Dialog auf **NEW...**



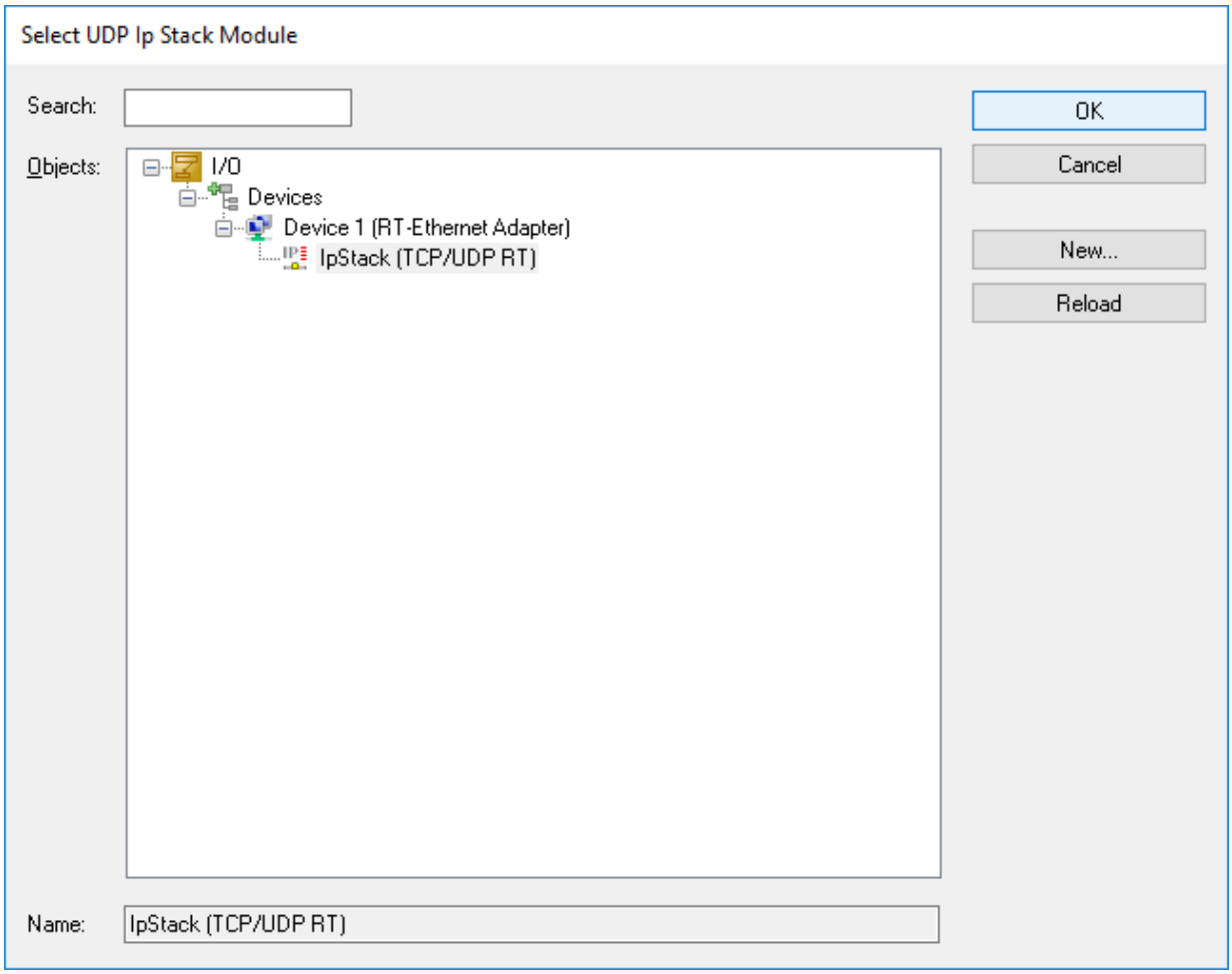
6. Wählen Sie im **Device Found At**-Dialog den Netzwerkport aus, an dem die Kamera angeschlossen ist und klicken Sie auf **OK**.
   Es werden nur Netzwerkports angezeigt, für die der TwinCAT RT-Ethernet Treiber installiert ist.



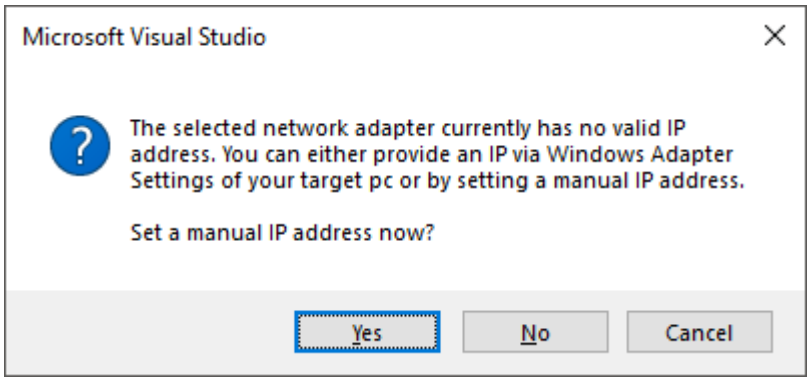
7. Das Netzwerk Gerät ist jetzt in der IO Konfiguration angelegt, wählen Sie es im **Select Network Device**-Dialog aus und klicken Sie auf OK. Damit wird ein IP Stack zum Gerät erstellt.



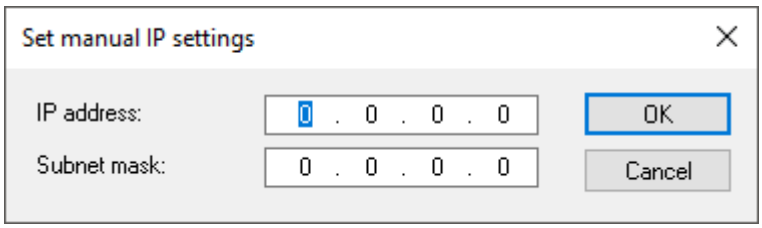
8. Wählen Sie den IP Stack aus und klicken Sie auf **OK**.



9. Ist dem Netzwerkport noch keine IPv4 Adresse zugewiesen, so beantworten Sie die Frage, ob Sie eine feste IP Adresse vergeben möchten.

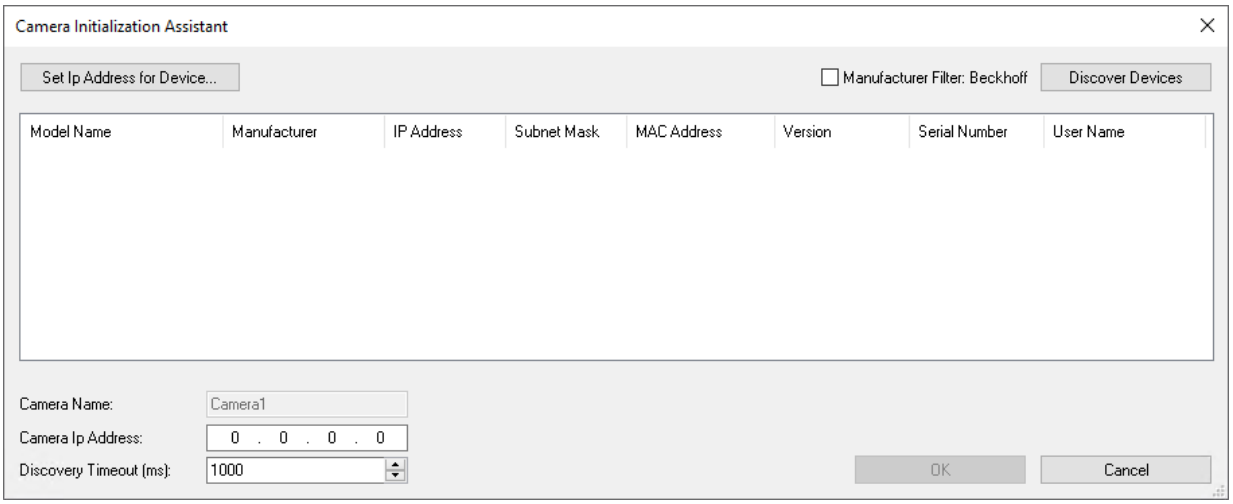


Wenn Ja, setzen Sie die IP-Adresse im nachfolgenden Dialog.

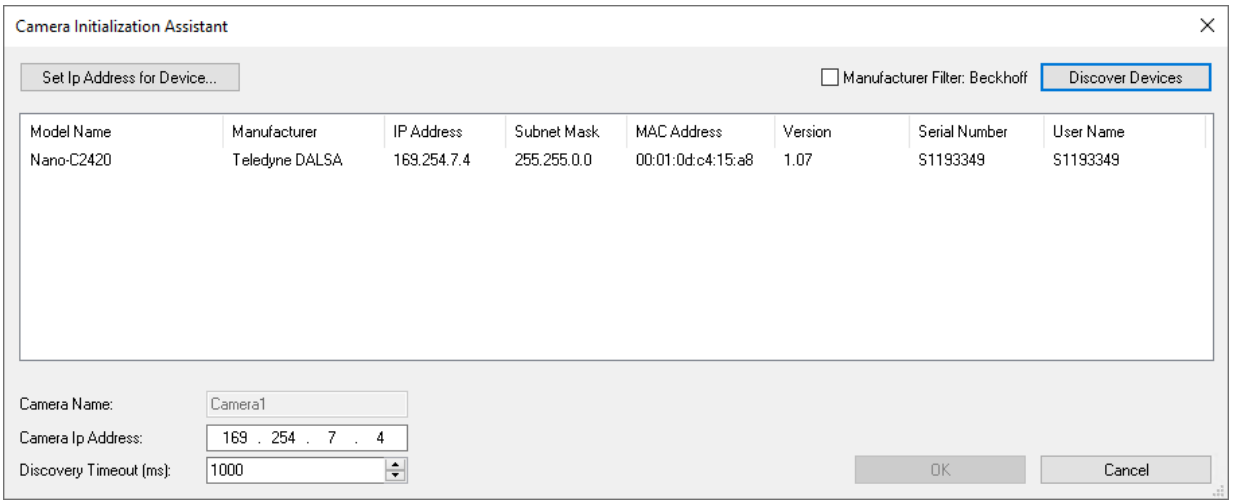


Die IP-Adresse und Subnetzmaske werden dann in der TwinCAT IP-Konfiguration des Netzwerkadapters gespeichert (siehe Netzwerkkonfiguration [▶ 61]).

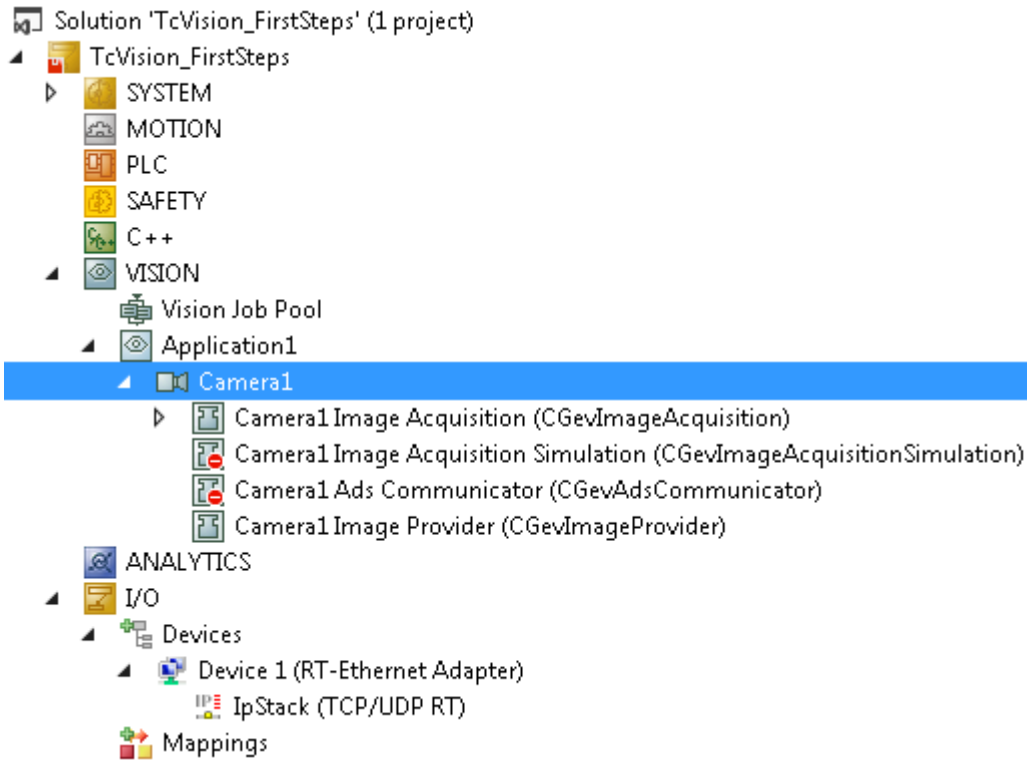
10. Im **Camera Initialization Assistant** klicken Sie auf **Discover Devices.** Der ausgewählte Netzwerkport wird nach angeschlossenen Kameras durchsucht.



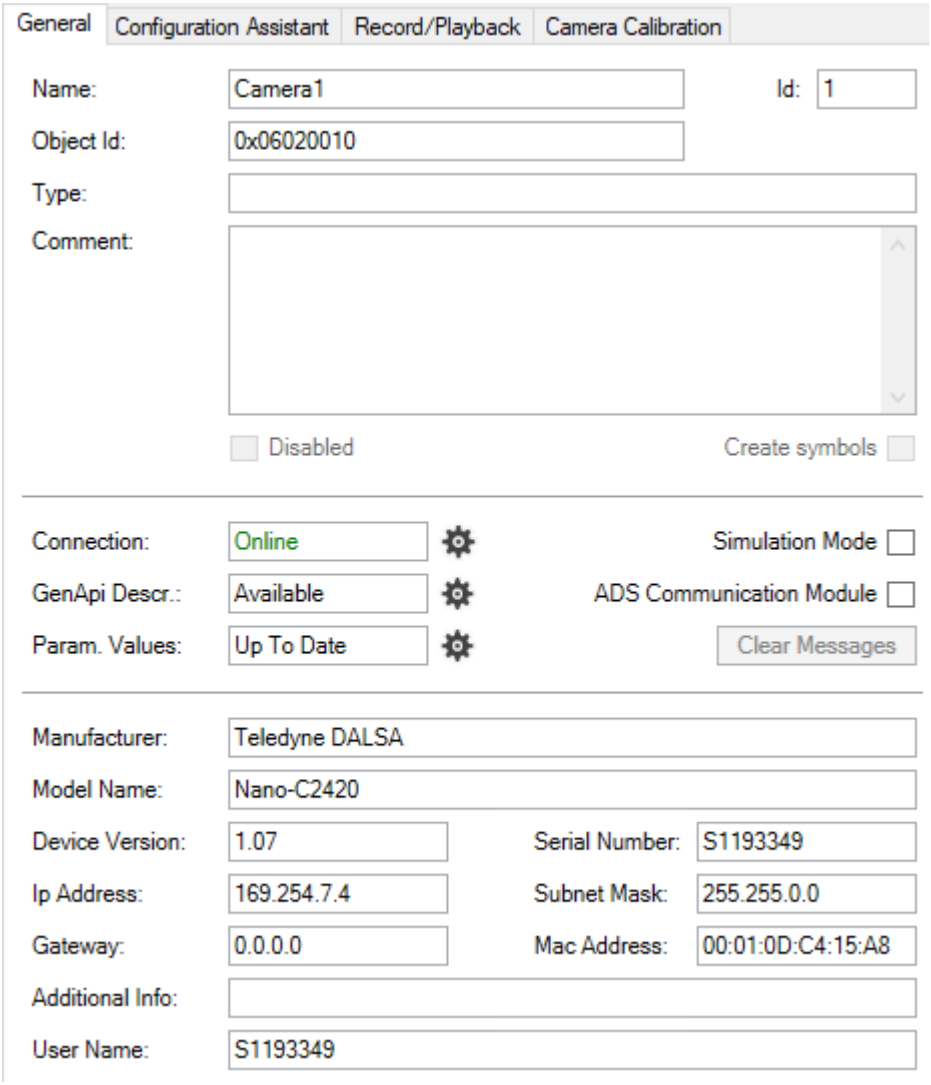
11. Alle gefundenen Kameras werden je nach Einstellung des Manufacturer Filter angezeigt. Wählen Sie die gewünschte Kamera aus und klicken Sie auf **OK**.



⇨ Die Kamera ist nun unterhalb der Applikation angelegt und kann verwendet werden.



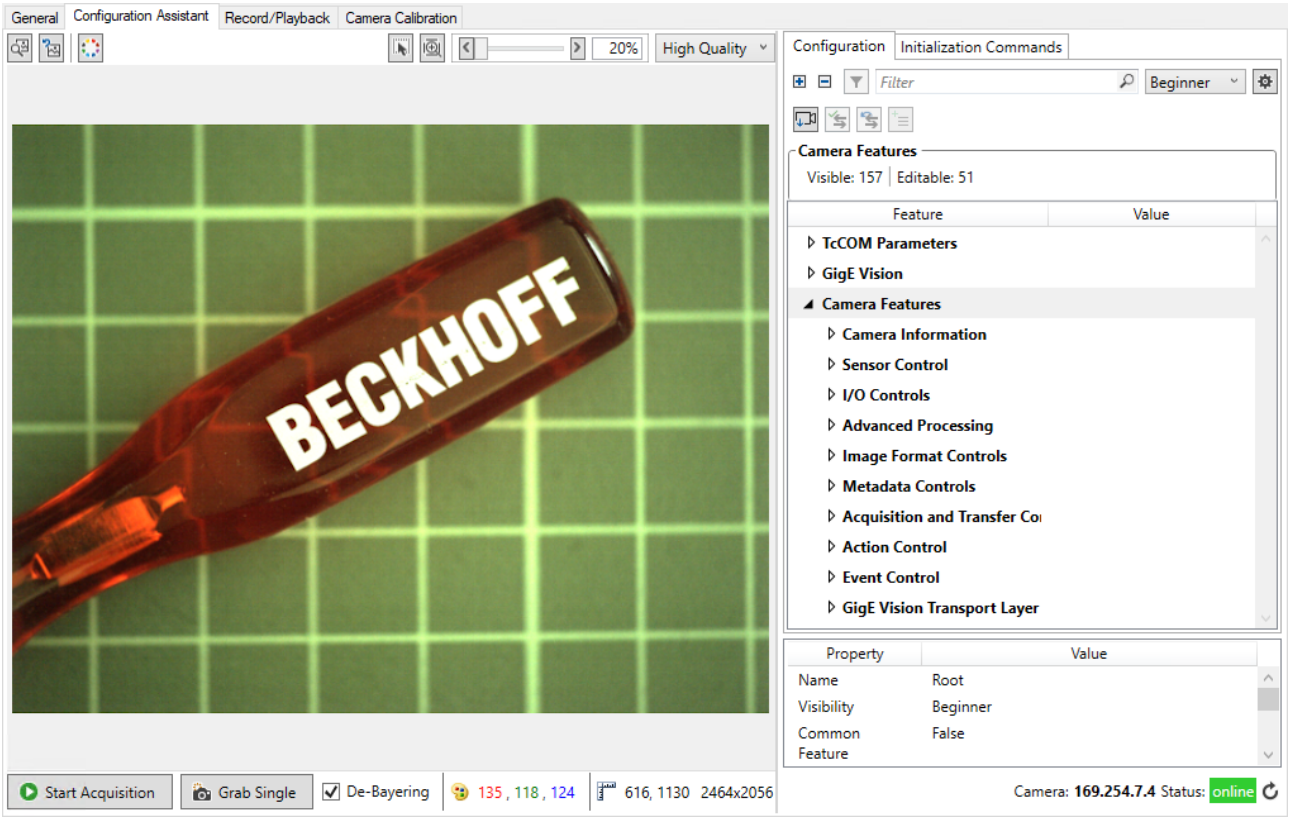
Mit einem Doppelklick auf den Kamera-Knoten [▶ 72] öffnet sich ein Fenster, in dem die Verbindung zur Kamera überprüft werden kann. Im General [▶ 74] sehen Sie allgemeine Informationen zur Kamera.



Im Configuration Assistant [▶ 79] können Sie über **Start Acquisition** das Live Bild der Kamera betrachten und Konfigurationseinstellungen an der Kamera vornehmen.

Sehen Sie dazu: Beispiele zur Kamera-Konfiguration [▶ 2932]

Wenn Sie die geänderten Parameter übernehmen und bei jedem Start an die Kamera schreiben möchten, erstellen Sie zum Abschluss die Initialization Commands [▶ 94].



Nächster Schritt

SPS-Projekt erstellen [▶ 47]

4.3 SPS-Projekt erstellen

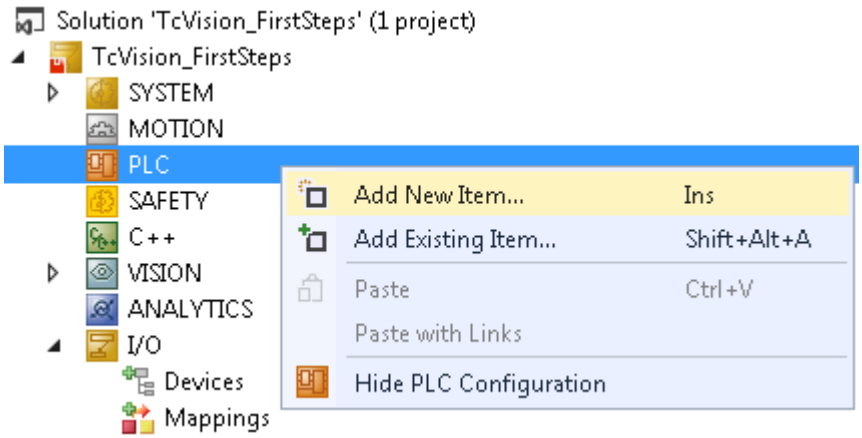
Video-Beispiel zur Erstellung eines SPS-Projekts mit TwinCAT Vision:

Video:: https://infosys.beckhoff.com/content/1031/tf7xxx_tc3_vision/Resources/5977084043.mp4

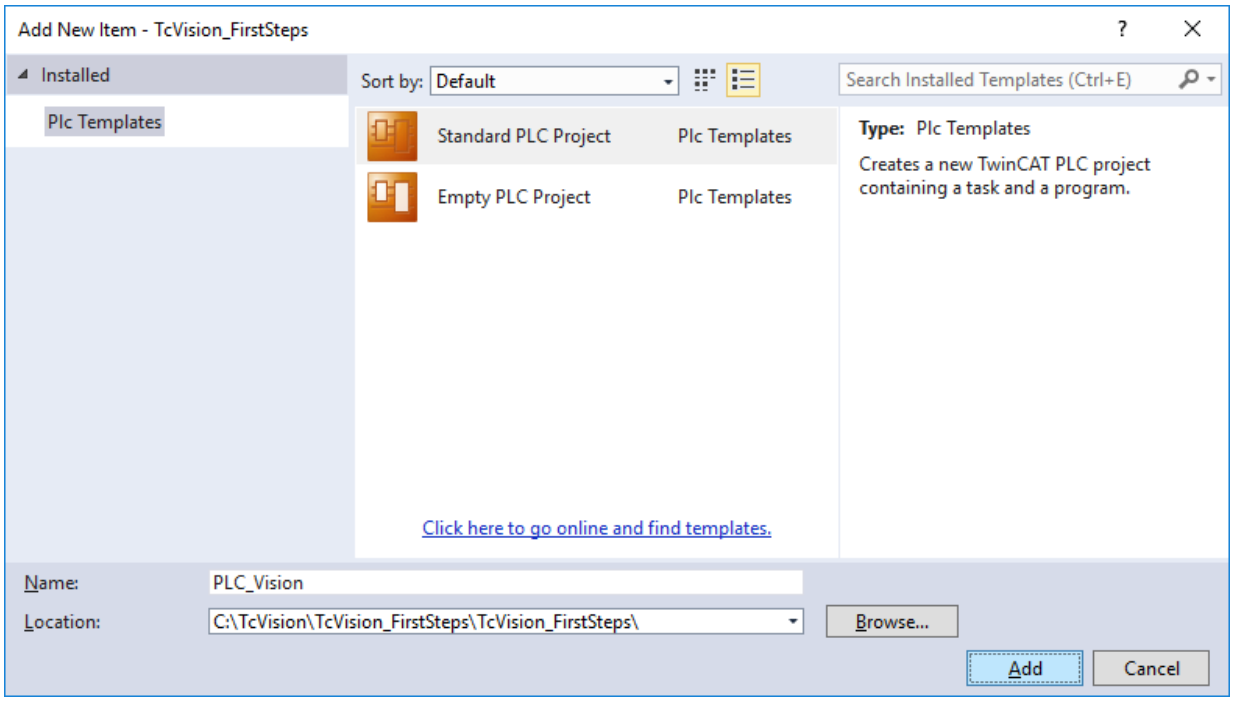
4.3.1 SPS-Projekt erstellen

Wie die Programmierung der allgemeinen Maschinensteuerung, erfolgt auch die Programmierung der Bildverarbeitung in der SPS. Dazu muss das TwinCAT-Projekt um ein SPS -Projekt erweitert werden.

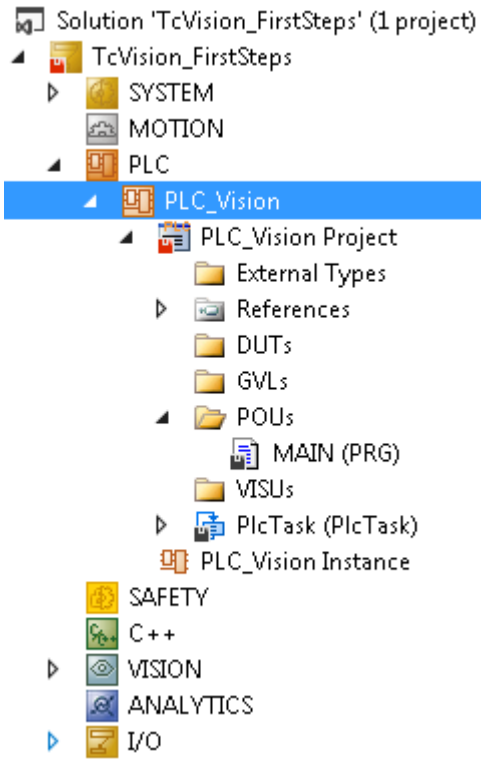
1. Machen Sie einen Rechtsklick auf den SPS-Knoten im Solution Explorer > **Add New Item...**



2. Wählen Sie den Namen und den Speicherort im Dialog **Add New Item**.



⇨ Das SPS-Projekt wurde erstellt.



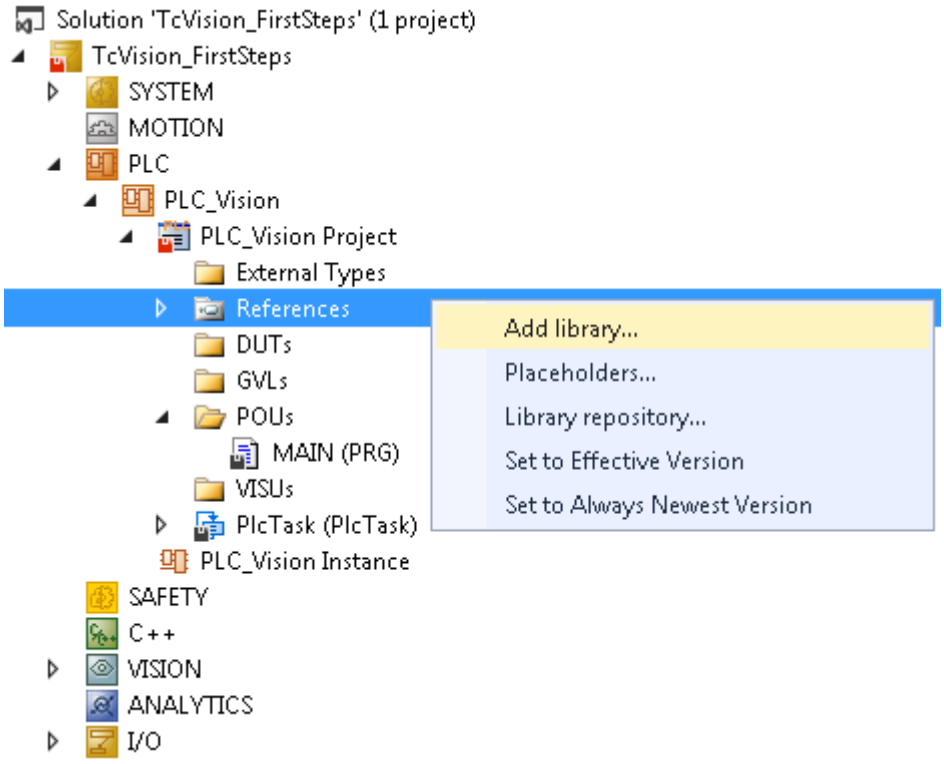
Nächster Schritt

SPS-Bibliothek einbinden [▶ 48]

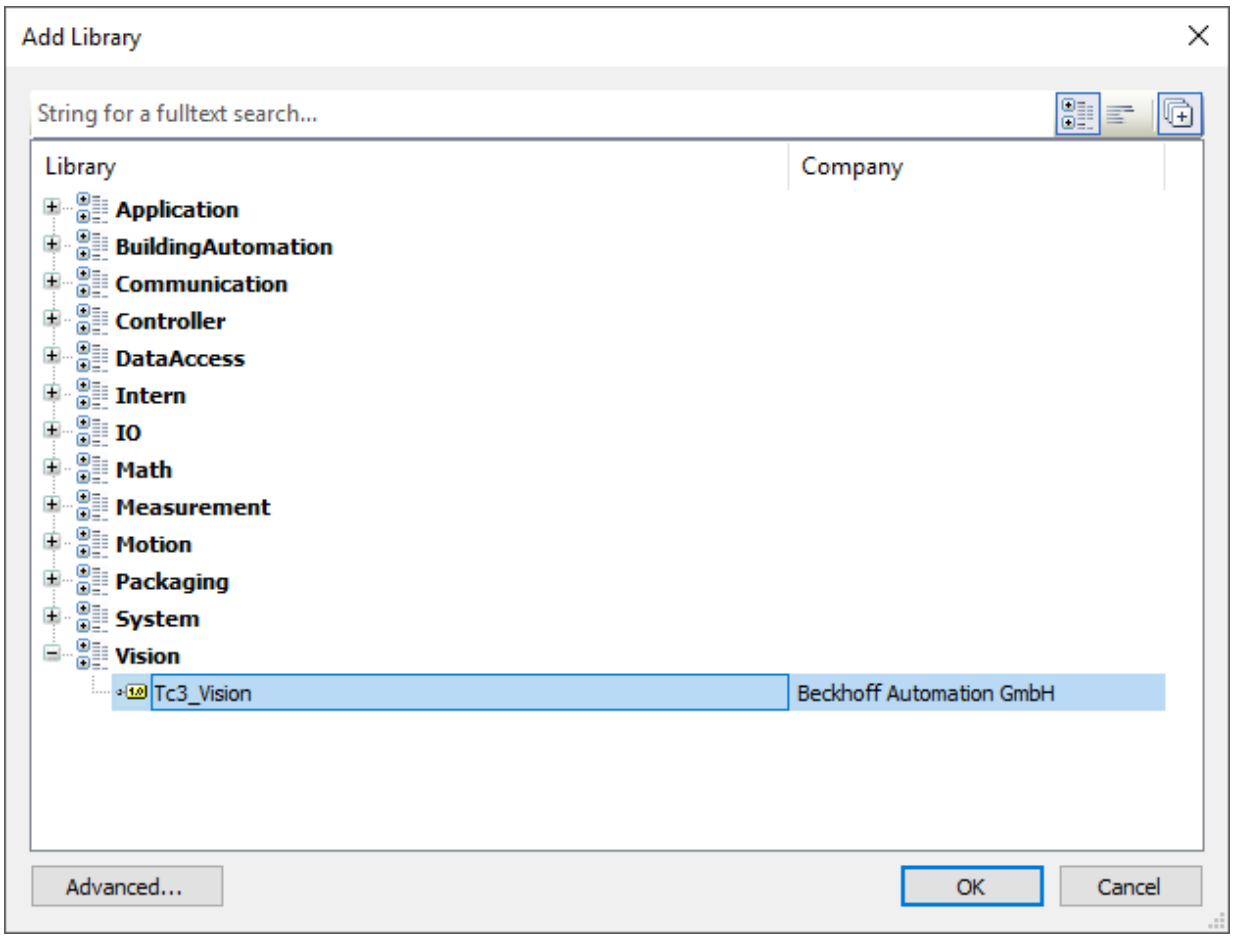
4.3.2 SPS-Bibliothek einbinden

Die Tc3_Vision-Bibliothek enthält die für die Bildverarbeitung notwendigen Datentypen, Funktionen und Funktionsbausteine.

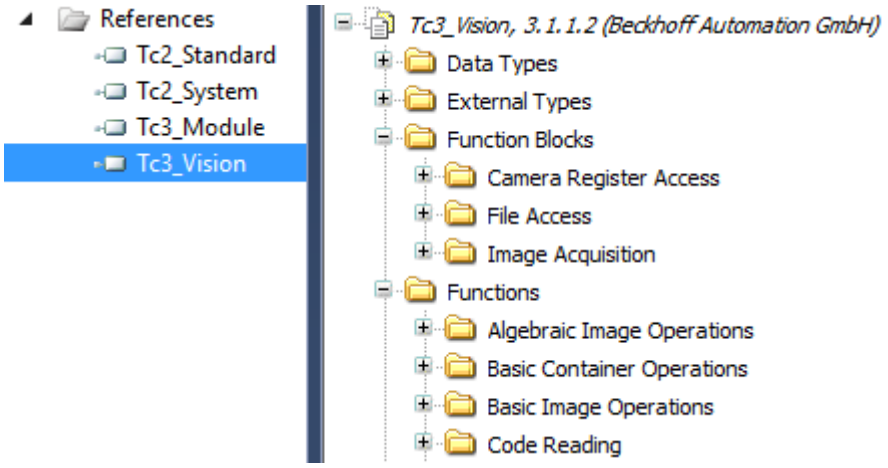
1. Um sie zum SPS-Projekt hinzuzufügen, machen Sie einen Rechtsklick auf **References** > **Add library...**



2. Wählen Sie die **Tc3_Vision** Bibliothek aus und bestätigen Sie mit **OK**.



⇨ Die Bibliothek ist nun hinzufügt und kann durch einen Doppelklick genauer betrachtet werden.



Nächster Schritt

SPS-Programm schreiben [▶ 50]

4.3.3 SPS-Programm schreiben

Zu einem einfachen SPS-Programm zur kontinuierlichen Bild-Akquise von einer Kamera oder einem File Source gehört Folgendes:

Variablen

```
PROGRAM MAIN
VAR
    hr                  :    HRESULT;
    fbCamera            :    FB_VN_SimpleCameraControl;
    eState              :    ETcVnCameraState;
    ipImageIn           :    ITcVnImage;
    ipImageInDisp       :    ITcVnDisplayableImage;
    nNewImageCounter    :    UINT;
END_VAR
```

Name	Typ	Beschreibung
hr	HRESULT	Alle Funktionen und viele Methoden der Tc3_Vision Bibliothek geben zur Fehlerbehandlung einen Ergebniscode vom Type HRESULT zurück. Mehr zu dem Konzept dahinter und der Bedeutung einzelner Codes in HRESULT [▶ 150].
fbCamera	FB_VN_SimpleCameraControl	Kameras und File Source Controls werden jeweils durch eine Funktionsbausteininstanz repräsentiert. Im einfachsten Fall ist dies eine Instanz vom FB_VN_SimpleCameraControl [▶ 1697] die die gemeinsamen Eigenschaften von Kamera und File Source Control abdeckt und daher mit beiden Gerätetypen verknüpft werden kann. Alternativ gibt es noch den FB_VN_GevCameraControl [▶ 1672] für Kameras und den FB_VN_FileSourceControl [▶ 1664] für File Source Controls.
eState	ETcVnCameraState	Jedes Vision-Gerät hat einen Zustand vom Typ ETcVnCameraState [▶ 179].
ipImageIn	ITcVnImage	Bilder zur Verarbeitung werden über Variablen vom Typ ITcVnImage [▶ 426] verwaltet. In diesem Beispiel erhält man über ipImageIn Zugriff auf das Eingangsbild der Kamera.
ipImageInDisp	ITcVnDisplayableImage	Bilder, die dargestellt werden sollen, werden über ITcVnDisplayableImages [▶ 426] verwaltet.

Programmcode mit Zustandsautomat

Jedes Vision-Gerät hat einen internen Zustand, wobei `TCVN_CS_INITIAL` der Anfangszustand ist. Von diesem kann das Gerät mit der Methode `StartAcquisition` in den `TCVN_CS_ACQUIRING` Zustand gebracht werden. In diesem können Bilder aufgenommen und übertragen werden. Liegt ein Bild vollständig im Speicher vor, kann es über `fbCamera.GetCurrentImage(ipImageIn)` abgerufen werden.

Zur Überprüfung, ob ein Bild vollständig empfangen wurde und abgerufen werden konnte, wird in der IF-Abfrage das `HRESULT` der Methode und der Pointer auf den Speicherbereich des Bildes auf ungleich 0 abgefragt. Danach kann mit der Analyse des Bildes begonnen werden. Zur Überprüfung und Anzeige dass ein Bild abgerufen werden konnte, wird beispielhaft die Zählervariable `nNewImageCounter` inkrementiert.

F_VN_TransformIntoDisplayableImage [▶ 835] verschiebt den Bild-Pointer anschließend in `ipImageInDisp`, um das Bild anzuzeigen. Das Bild kann danach nicht mehr über `ipImageIn` abgerufen werden, alternativ kann mit F_VN_CopyIntoDisplayableImage [▶ 804] auch eine Kopie erzeugt werden, um das Bild sowohl anzeigen als auch weiterbearbeiten zu können.

```
eState := fbCamera.GetState();

IF eState = TCVN_CS_ERROR THEN
    hr := fbCamera.Reset();

ELSIF eState < TCVN_CS_ACQUIRING THEN
    hr := fbCamera.StartAcquisition();

ELSIF eState = TCVN_CS_ACQUIRING THEN
    hr := fbCamera.GetCurrentImage(ipImageIn);

    // Check if new Image was received
    IF SUCCEEDED(hr) AND ipImageIn <> 0 THEN
        nNewImageCounter := nNewImageCounter + 1;

        // Place to call vision algorithms
        hr := F_VN_TransformIntoDisplayableImage (ipImageIn, ipImageInDisp, hr);
    END_IF
END_IF
```

Weitere Infos zum Zustandsautomat finden Sie imAPI-Kapitel Image Acquisition [▶ 1662].

Weiterführende Informationen

- API-Referenz [▶ 146]
- Funktionsblöcke zur Bild-Akquise [▶ 1662]
- Bilder [▶ 159]

Nächster Schritt

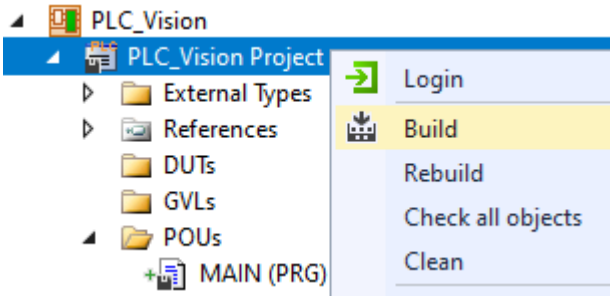
Funktionsbaustein initialisieren [▶ 51]

4.3.4 Funktionsbaustein initialisieren

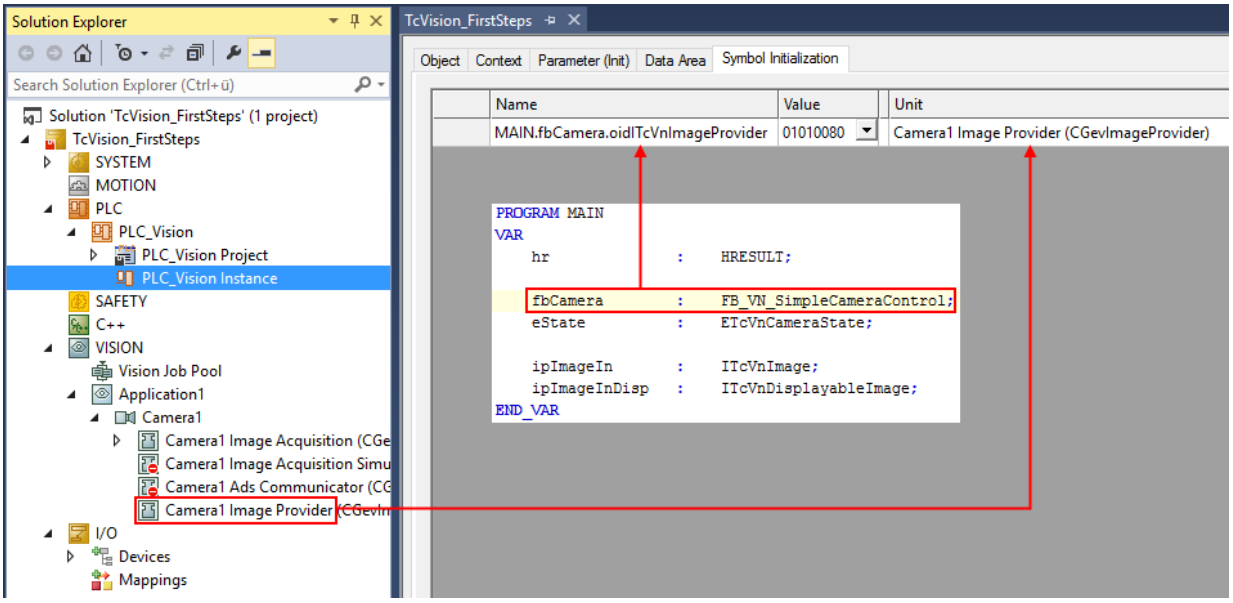
Nach dem Konfigurieren eines Vision-Gerätes und dem Anlegen eines SPS-Programms, muss das Vision-Gerät nun mit entsprechenden Kamera-Variablen in der SPS verknüpft werden.

Die Instanz `fbCamera` des Funktionsbausteins FB_VN_SimpleCameraControl [▶ 1697] aus dem SPS-Programm muss mit dem Image Provider des Vision-Geräts verknüpft werden. Dazu:

1. Bauen Sie das SPS-Programm [▶ 50], indem Sie einen Rechtsklick auf das SPS-Projekt machen und **Build** auswählen. Stellen Sie sicher, dass dabei keine Fehler auftreten. Nur wenn es fehlerfrei gebaut wurde, erscheint in der SPS-Instanz der Reiter **Symbol Initialization** und darunter die Instanz `fbCamera`.



2. Machen Sie einen Doppelklick auf die **Instanz des SPS-Projekts** und öffnen Sie den Karteireiter **Symbol Initialization**.
3. Weisen Sie dem Symbol `MAIN.fbCamera.oidITcVnImageProvider` den ImageProvider Ihres Vision-Geräts über das Dropdown in der Value-Spalte zu. Im Bild ist dies für ein Kamera Objekt dargestellt, für File Source Objekte funktioniert dies genauso.



Nächster Schritt

TwinCAT-Projekt aktivieren und SPS-Projekt starten [▶ 52]

4.4 Ausführen

Video-Beispiel zur Ausführung eines TwinCAT Vision SPS-Projekts:

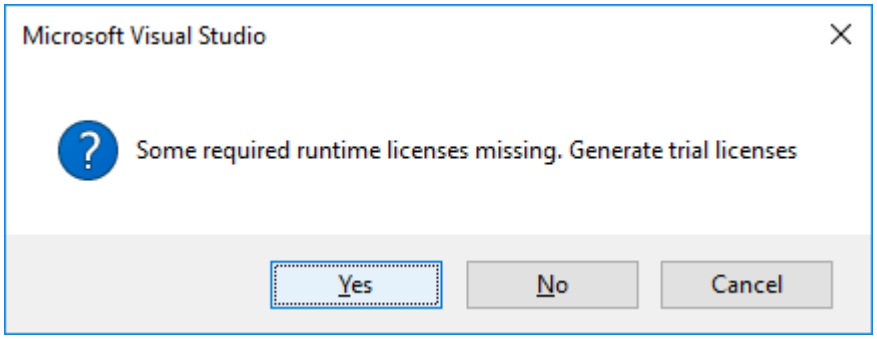
Video:: https://infosys.beckhoff.com/content/1031/tf7xxx_tc3_vision/Resources/5977086347.mp4

4.4.1 TwinCAT-Projekt aktivieren und SPS-Projekt starten

Sie können das Projekt mit den folgenden Schritten ausführen:

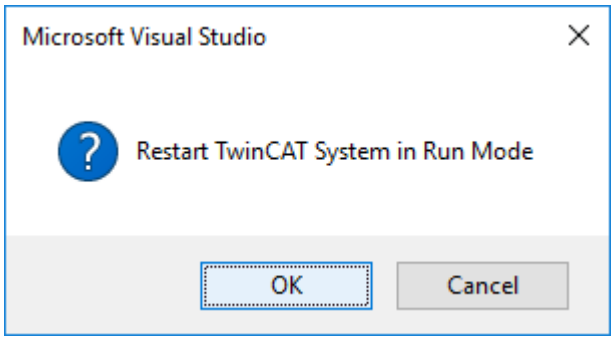
1. Aktivieren Sie die Konfiguration über oder per Visual Studio Menü > **TwinCAT** > **Activate Configuration**.

2. Sind noch keine Lizenzen auf dem Zielsystem aktiviert, werden Sie über nachfolgenden Dialog darauf hingewiesen. Für einen ersten Test können Sie mit **Yes** und dem Eintragen eines Security Codes eine Test-Lizenz erzeugen.



3. Anschließend werden Sie gefragt, ob Sie TwinCAT neustarten wollen. Dies bestätigen Sie mit **Yes**.

   Alternativ kann über oder per Visual Studio Menü > **TwinCAT** > **Restart TwinCAT System** der Neustart ausgelöst werden.



4. Ist kein Autostart der SPS eingestellt, so starten Sie sie separat. Dazu über oder per Visual Studio Menü > **SPS** > **Einloggen** die SPS einloggen.

5. Anschließend starten Sie über oder per Visual Studio Menü > **PLC** > **Start**.

Nächster Schritt

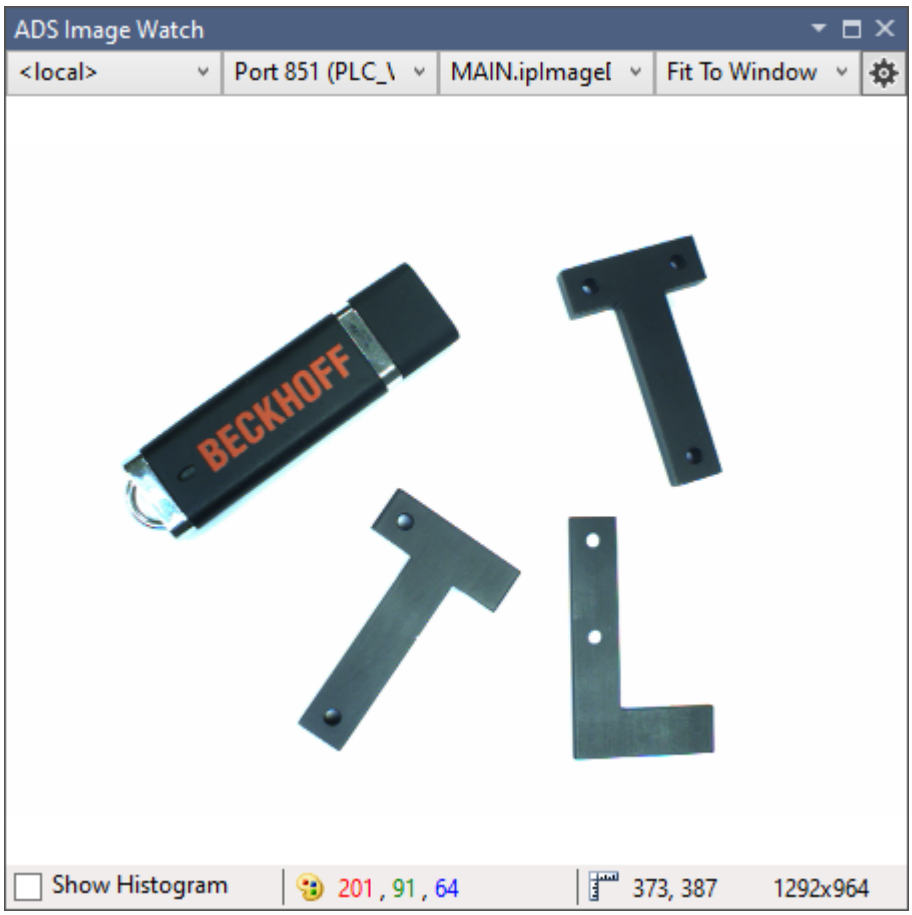
Bildanzeige im ADS Image Watch [▶ 53]

4.4.2 Bildanzeige im ADS Image Watch

Bilder vom Typ `ITcVnDisplayableImage` in der SPS können im sogenannten ADS Image Watch dargestellt werden. Sie finden das Fenster im Visual Studio Menü **TwinCAT > Windows > ADS Image Watch.**

Zur Auswahl des anzuzeigenden Bildes stellen Sie Folgendes von links nach rechts ein:

- Zielsystem
- Port
- ADS-Symbol des Bildes
- Skalierung des Bildes

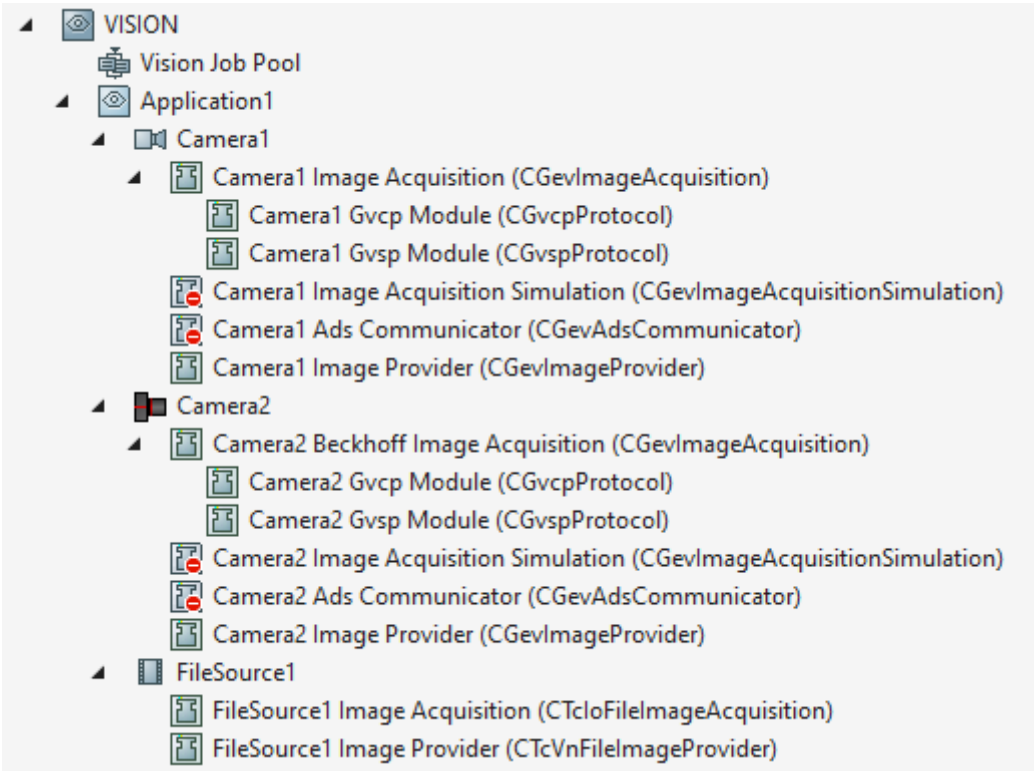


Zusätzlich zum Bildinhalt können im ADS Image Watch folgende Infos ausgelesen werden:

- Histogramm des Bildes
- Farbwerte/Intensitätswerte der aktuellen Cursor-Position
- Pixel-Koordinaten der aktuellen Cursor-Position
- Bildgröße

5 Entwicklungsumgebung

TwinCAT Vision ist vollständig in die TwinCAT 3 Entwicklungsumgebung integriert. Im Wesentlichen erweitert TwinCAT Vision das TwinCAT System um einen weiteren Knoten, untern dem die Vision-Komponenten angelegt und konfiguriert werden können. Dabei werden folgende Konfigurationselemente verwendet:



- Der VISION Knoten [▶ 68] bietet allgemeine Konfigurationsmöglichkeiten, die alle Vision-Komponenten auf dem System betreffen, und enthält sämtliche weiteren Konfigurationselemente.
- Im Vision Job Pool [▶ 140] können verfügbare Job-Tasks zur parallelisierten Ausführung von Vision-Funktionen festgelegt werden. Jedes System besitzt einen einzigen Vision Job Pool.
- Ein Application Knoten [▶ 71] kapselt mehrere Vision-Geräte (GigE Vision Kameras und File Sources) zu einer applikativen Einheit. Die gesamte Konfiguration einer Applikation kann exportiert und importiert werden. Jedes TwinCAT Projekt kann mehrere Applikationen haben.
- Ein GigE Vision Kamera Knoten [▶ 72] enthält die Konfiguration einer physischen GigE Vision Kamera. Jede Applikation kann mehrere GigE Vision Kamera Knoten haben.
- Ein Beckhoff-Kamera-Knoten [▶ 135] enthält die Konfiguration einer physischen Beckhoff-Kamera. Jede Applikation kann mehrere Beckhoff-Kamera-Knoten haben.
- Ein File Source Knoten [▶ 135] enthält die Konfiguration für eine Schnittstelle zwischen Bildern auf der Festplatte des Zielsystems und einem Bild-Akquisitions-Modul innerhalb der SPS. Jede Applikation kann mehrere File Source Knoten haben.

Kommunikation mit System-Komponenten

GigE Vision-Kamera-Knoten, Beckhoff-Kamera-Knoten und File Source-Knoten enthalten jeweils TcCOM-Module zur Kommunikation mit verschiedenen System-Komponenten.

Des Weiteren existiert in der Entwicklungsumgebung das Fensterelement ADS Image Watch [▶ 141] zur Anzeige von Bildern aus der SPS per ADS. Außerdem ist für die korrekte Funktionsweise der Vision-Komponenten eine generelle Systemkonfiguration [▶ 56] notwendig.

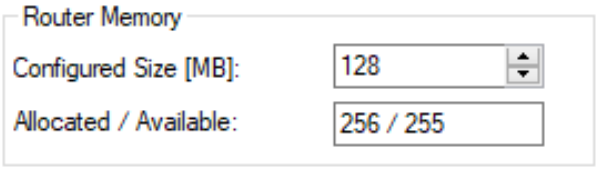
Um Bilder aus der SPS live in einer TwinCAT HMI anzeigen zu können, steht eine entsprechende HMI-Erweiterung [▶ 2784] zur Verfügung.

5.1 Systemkonfiguration

Die folgenden Abschnitte beschreiben die TwinCAT Vision spezifischen Einstellungen, die system- und projektbezogen durchgeführt oder geprüft werden sollten.

5.1.1 Router-Speicher

Bilderverarbeitungsapplikationen benötigen im Vergleich zu herkömmlichen Steuerungen viel Speicher, da schon ein einzelnes Bild mehrere Mega-Byte Daten enthalten kann. In TwinCAT Vision werden sämtliche dynamischen Daten (siehe Interface Pointer [▶ 146]) im Router-Speicher allokiert. Aus diesem Grund ist es besonders wichtig, die Größe des Router-Speichers und mit der TwinCAT 3.1 4026 den Core Memory [▶ 58] entsprechend auszulegen. Die Einstellung kann in der TwinCAT Systemkonfiguration unter dem Real-Time Knoten im Reiter **Einstellungen** vorgenommen werden:



Configured Size	Einstellung des für das aktuelle Projekt benötigten Router-Speichers.
Allocated	Aktuell auf dem System allokierter Router-Speicher.
Available	Teil des aktuell auf dem System allokierten Router-Speichers, der noch frei verfügbar ist. (Der Rest ist innerhalb von TwinCAT schon benutzt.) **Wenn der verfügbare Speicher sich nach einem TwinCAT-Neustart nicht vollständig wiederherstellt, deutet dies auf ein Speicher-Leck hin. Starten Sie dann ihr System neu und überprüfen Sie Ihr Programm hinsichtlich Speicher-Lecks.**

Die maximal einstellbare Größe des Router-Speichers in TwinCAT 3.1 4024 beträgt 1024 MB. Ab TwinCAT 3.1 4026 hängt die verwendbare Größe im Wesentlichen vom verfügbaren Arbeitsspeicher ab und ist auf 65535 MB begrenzt. Die tatsächlich benötigte Speichergröße hängt unter anderem von der Nutzung von dynamischem Speicher (z. B. bei Bildern [▶ 159] und Containern [▶ 163]) im SPS-Programm ab.

HINWEIS

Größe des Router-Speichers

Bei der Konfiguration des Router-Speichers ist zu beachten, dass der Router-Speicher aus dem Arbeitsspeicher bezogen wird und somit dem Betriebssystem nicht mehr zur Verfügung steht. Daher wird empfohlen, die eine Größe von 1024MB erst ab einer Arbeitsspeicher-Größe von 4GB zu konfigurieren. Bei fremden Projekten überprüfen Sie vor dem Aktivieren der Konfiguration immer die Router-Speicher-Einstellung und reduzieren Sie sie gegebenenfalls.

Abschätzung der benötigten Speicher-Größe

Die benötigte Größe des Router-Speichers ist von vielen Faktoren abhängig. Eine ungefähre Abschätzung für den Vision-Teil des TwinCAT-Projekts kann mittels Berechnung des (durch Interface Pointer [▶ 146]) dynamisch allokierten Speichers getroffen werden.

Die Berechnung des Speicherplatzes `M` (in Byte) eines Bildes kann dabei durch die Einbeziehung von Bildgröße (Breite `w` und Höhe `h`), Anzahl `C` der Kanäle und der Byte-Anzahl des Element-Typs erfolgen: `M = w * h * C * sizeof(ElementType).`

Für einen Container vom Typ String oder Vektor kann die Berechnung des Speicherplatzes `M` (in Byte) durch die Anzahl an allokierten Elementen `N` und der Byte-Anzahl des Element-Typs erfolgen: `M = N * sizeof(ElementType).`

Es muss darauf geachtet werden, wie die Daten kopiert und freigegeben werden. Wenn ein Bild kopiert oder ein neues erzeugt wird, muss natürlich der Speicherplatz beider Instanzen einberechnet werden. Darüber hinaus muss der Speicherbedarf des Routers selbst und anderer Projektteile einberechnet werden. Beim Abrufen der Bilder oder anderer Daten per ADS werden diese kopiert, daher muss dafür ebenfalls zusätzlicher Speicherplatz berücksichtigt werden.

HINWEIS

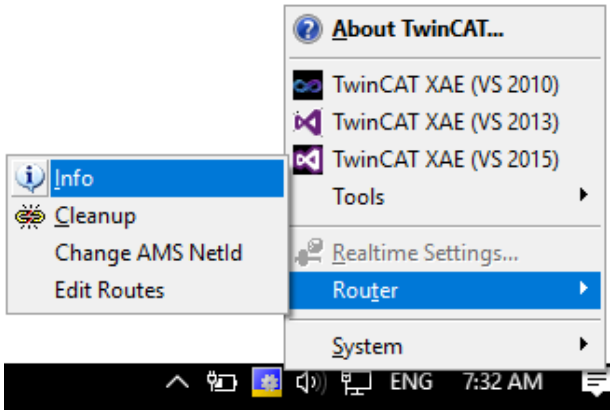
Speicherwert mit Sicherheitsfaktor

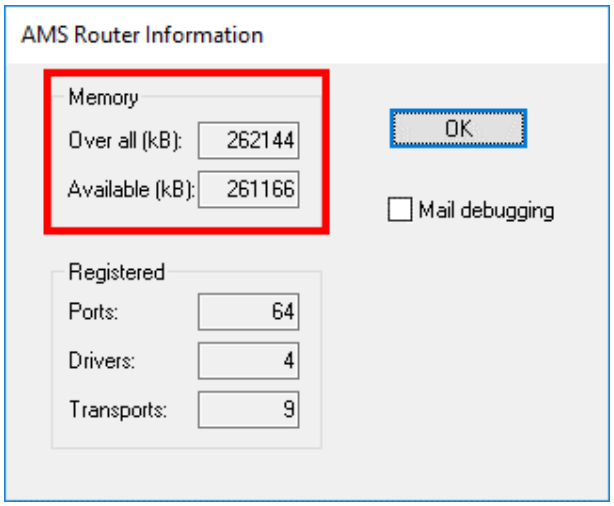
Erweitern Sie den berechneten Speicherwert immer mit einem Sicherheitsfaktor. Berücksichtigen Sie dabei ebenfalls den Speicherbedarf der anderen Module des Projekts.

Um nicht unnötig viel Platz im Router-Speicher zu verbrauchen, sollten Sie darauf achten, keine unnötigen, tiefe Kopien zu machen und nicht mehr benötigte Objekte wieder freizugeben (siehe dazu Interface Pointer [▶ 146]). Um herausfinden, wie viel Speicher tatsächlich verbraucht wird, kann der Router-Speicher live wie folgt überwacht werden.

Überwachung des Router-Speichers

Auch wenn die Einstellung des Router-Speichers im Projekt erfolgt, handelt es sich dabei um eine system-weite Einstellung. Daher können der im Projekt eingestellte und der auf dem System aktuell verwendete Router-Speicher auf dem Zielsystem eingesehen werden. Tätigen Sie dazu einen Rechtsklick auf das TwinCAT-Symbol in der Windows Taskleiste und wählen Sie **Router > Info**.





HINWEIS

Verringerung des verfügbaren Router Speichers

Durch Exceptions oder nicht freigegebene Interface Pointer kann es dazu kommen, dass der verfügbare Router Speicher verringert wird. Daher sollte der Speicher während der Entwicklung, insbesondere im Fehlerfall, überprüft werden. Falls nicht mehr genug Speicher zur Verfügung steht, starten Sie das komplette System neu und überprüfen Sie im Anschluss Ihr Programm.

5.1.2 Core Memory

Ab TwinCAT Version 3.1 4026 ist es möglich, zusätzlich zum Router-Speicher, für jeden Echtzeitkern einen separaten Speicherbereich zu reservieren. Alle Speicheranfragen aus diesem Kern werden zuerst von diesem Speicher bearbeitet und nur wenn dieser nicht ausreichend ist, wird Speicher vom Router angefragt. Weitere Einstellmöglichkeiten und Details finden Sie in der TwinCAT Engineering Dokumentation unter den Echtzeiteinstellungen.

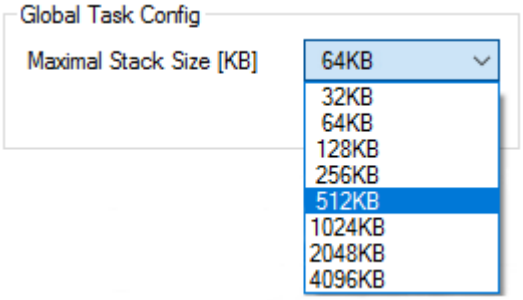
Bei der Verwendung von TwinCAT Vision werden in der Regel viele und teilweise auch größere dynamischen Daten wie z.B. Bilder verwendet, daher empfiehlt es sich, den Core Memory zu verwenden und entsprechend groß einzustellen. Dadurch werden die Zugriffe auf den allgemeinen Router Speicher reduziert, was bei parallelen Speicherzugriffen zu schnelleren Speicherallokationen führt.

Core	RT-Core	Base Time	Core Limit	Latency Warning	Core Memory	Core Frequency
0 (Shared)	☐					Base Frequency
1 (Shared)	☐					Base Frequency
2 (Shared)	☐					Base Frequency
3 (Shared)	☑	1 ms	80 %	(none)	512 KB with 2 KB limit	Base Frequency
4 (Shared)	☑ Default	1 ms	80 %	(none)	512 KB with 2 KB limit	Base Frequency
5 (Shared)	☐					Base Frequency
6 (Isolated)	☑	1 ms	100 %	(none)	16 MB	Base Frequency
7 (Isolated)	☑	1 ms	100 %	(none)	64 MB	Base Frequency

5.1.3 Stack-Größe

Die Defaultgröße von 64 KB reicht für Bilderverarbeitungsapplikationen in der Regel aus. Bei der Verwendung von größeren Bildformaten oder speziellen Funktionen wie z. B. der F_VN_Clahe [▶ 1268] ist dieser Wert zu klein und die Stackgröße muss erhöht werden. Dann empfiehlt es sich direkt 512 KB oder ggf. größer zu verwenden.

Die Einstellung kann in der TwinCAT Systemkonfiguration unter dem Real-Time Knoten im Reiter **Einstellungen** vorgenommen werden:



5.1.4 Netzwerkadapter

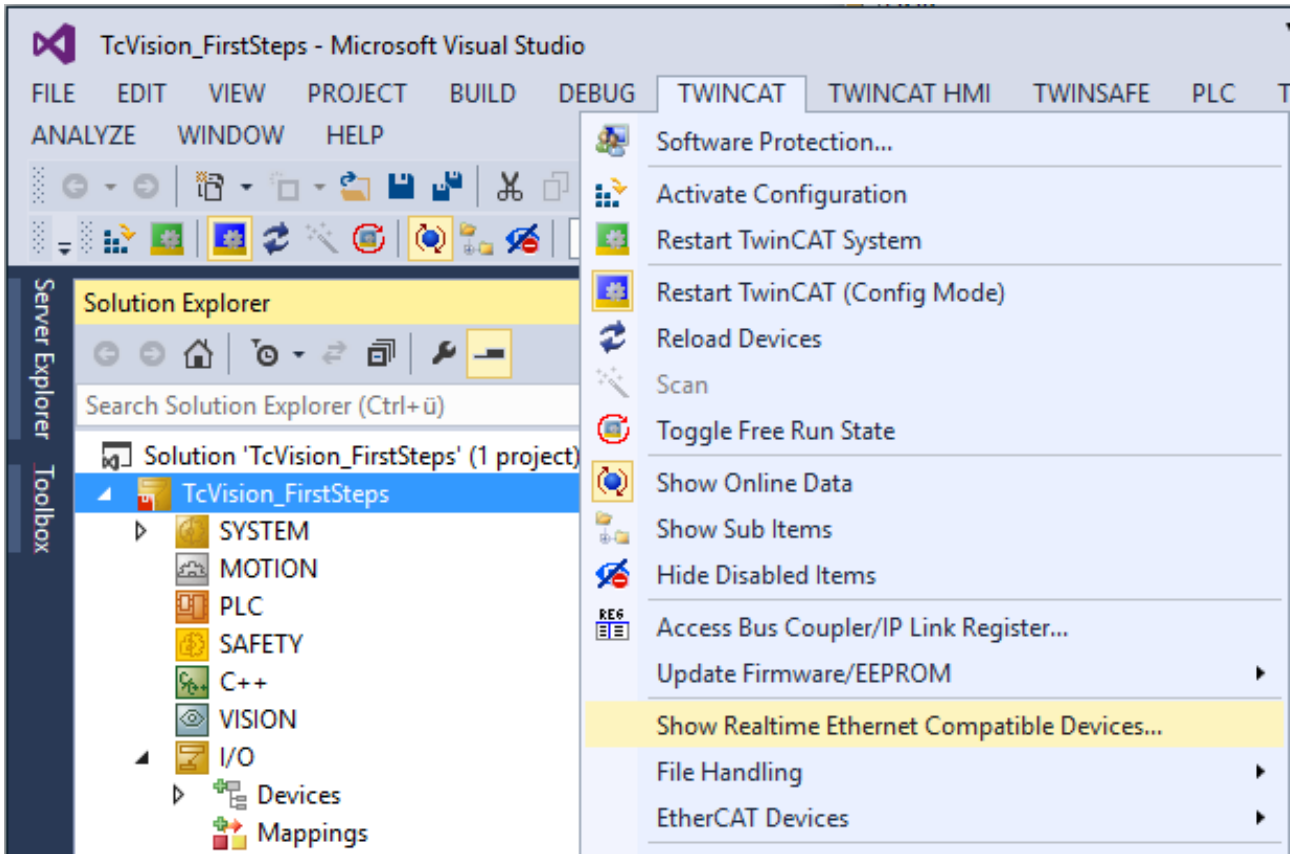
Bei der Auswahl, Konfiguration und Verwendung eines Netzwerk-Adapters gibt es mehrere Dinge zu beachten, die nachfolgend einzeln aufgelistet und beschrieben werden:

- Zunächst muss ein geeigneter Netzwerk-Adapter ausgewählt werden, der von unserem TwinCAT Realtime Ethernet Treiber unterstützt wird, so dass der Teiber installiert und im Echtzeitbetrieb verwendet werden kann.
- Danach sollten für diesen Netzwerk-Adapter alle sonstigen, nicht benötigen Netzwerkprotokolle deaktiviert werden, um Störungen in der Kommunikation auszuschließen.
- Nach dem erstmaligen Einrichten einer Kamera sollte eine persistente IP-Adresse im IP-Stack des TwinCAT Projekts eingestellt werden.
- Falls mehrere Kameras an dem gleichen Port des Netzwerk-Adapters betrieben werden sollen, müssen weitere Einstellungen im IP-Stack vorgenommen werden.

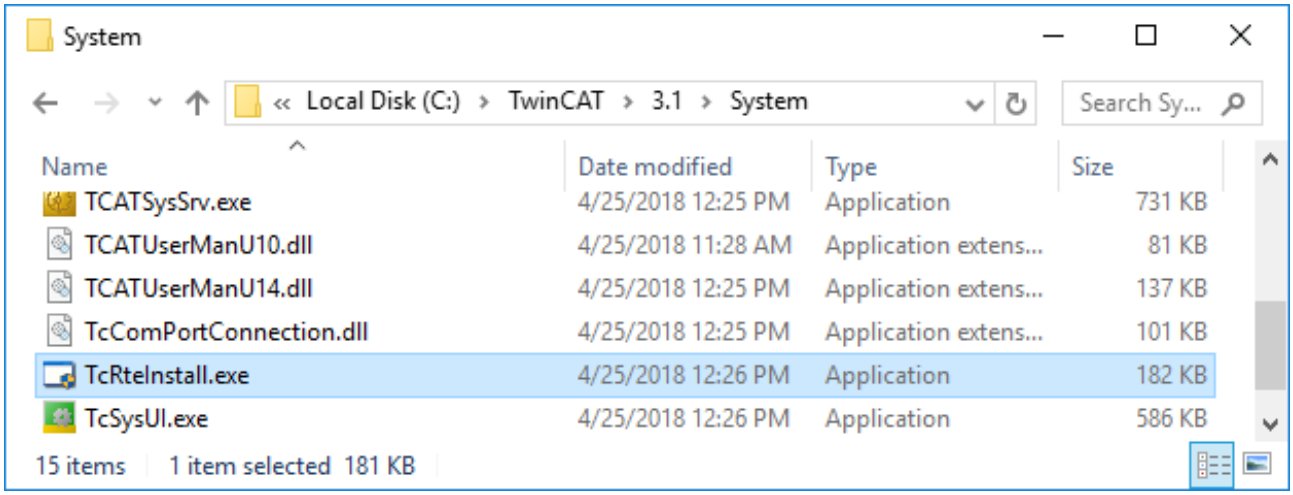
- Zum Abschluss finden Sie noch ein paar Hinweise zu den Diagnosemöglichkeiten.

TwinCAT RT-Ethernet Adapters

Analog zur Anbindung von EtherCAT-Geräten muss für die Verbindung zu GigE Vision Geräten der TwinCAT Realtime Ethernet Treiber auf dem entsprechenden Netzwerkadapter installiert werden. Zum Öffnen eines entsprechenden Installationsdialogs klicken Sie im Visual Studio Menü auf **TWINCAT > Show Realtime Ethernet Compatible Devices**.



Alternativ können Sie die Datei *C:\TwinCAT\3.1\System\TcRteInstall.exe* auf dem Zielsystem aufrufen; dies öffnet ebenfalls den entsprechenden Installationsassistenten.

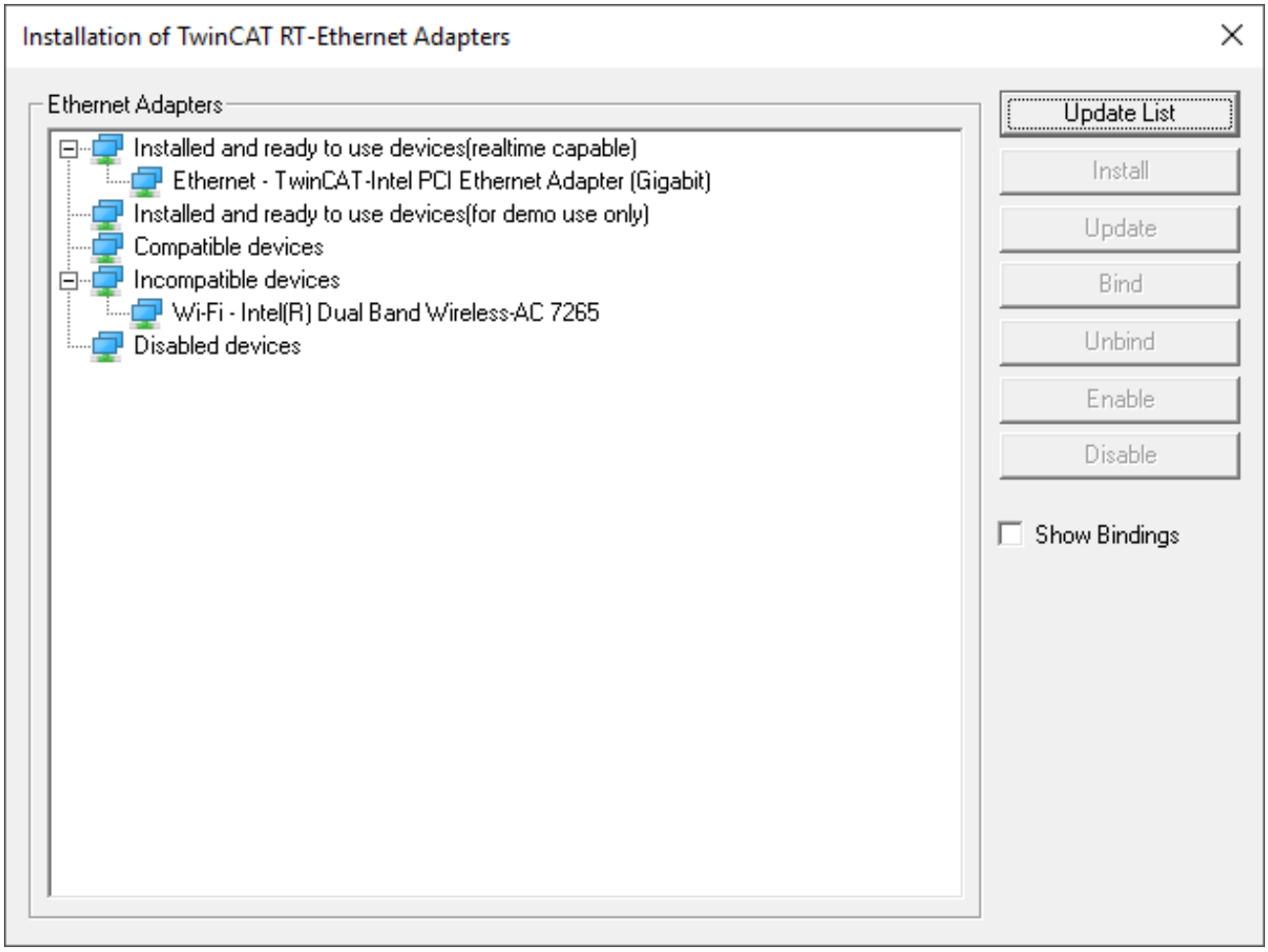


Stellen Sie im Installationsassistenten sicher, dass der entsprechende Netzwerk-Adapter unter **Installed and ready to use devices (realtime capable)** gelistet ist. Falls dies nicht der Fall ist, wählen Sie einen geeigneten Adapter aus der Liste **Compatible devices** aus und klicken Sie auf **Install**.

HINWEIS

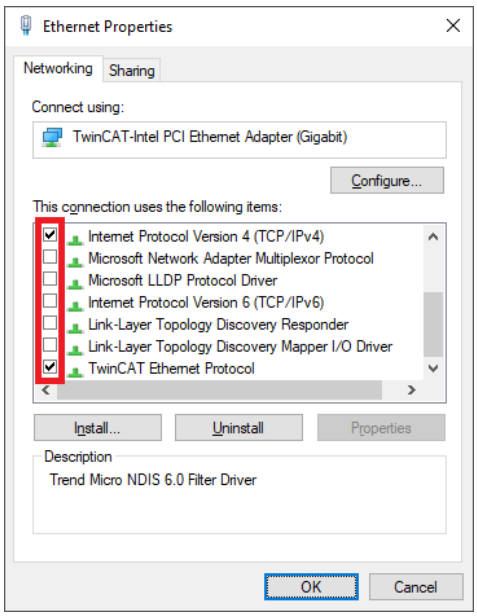
For demo use only

Dieser Status ist für eine stabile Verbindung und den Betrieb von GigE Vision Geräten nicht ausreichend. Falls der Netzwerk-Adapter nach der Installation nur unter dieser Gruppe gelistet ist, muss ein anderer Adapter verwendet werden.



Sonstige Netzwerkprotokolle

Es wird empfohlen, die Nutzung des Netzwerkadapters auf das TwinCAT Ethernet Protokoll zu beschränken und eine persistente IP-Adresse über den IP-Stack im TwinCAT Projekt zu vergeben. Denn wenn mehrere Protokolle Zugriff auf den Netzwerkadapter haben, können diese Pakete senden und Störungen bei der Kamera-Kommunikation verursachen. Die Einstellung dazu finden Sie in den Windows-Eigenschaften des Netzwerkadapters.



Das Internet Protokoll Version 4 (IPv4) wird nur dann benötigt, wenn DHCP zum Einrichten der Kameraverbindung verwendet werden soll. Beim erstmaligen Einrichten einer Kamera kann die automatische Vergabe einer Link-Local-Adresse (LLA) hilfreich sein. In allen anderen Fällen bzw. nach Abschluss der initialen Einstellungen können Sie das IPv4-Protokoll ebenfalls deaktivieren und die IP-Adresse manuell in TwinCAT im IP-Stack [▶ 61] konfigurieren.

HINWEIS

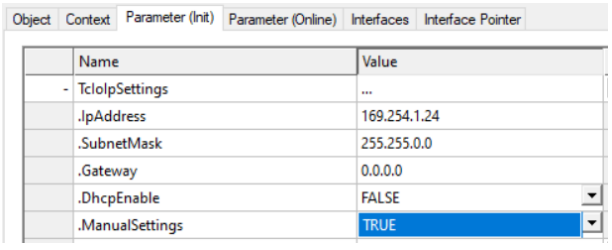
Deaktivierung sonstiger Netzwerkprotokolle

Wenn möglich, deaktivieren Sie auf Ihrem Ethernet Adapter alle Netzwerkprotokolle außer dem TwinCAT Ethernet Protocol. Anderenfalls kann es zu Fehlern und Verzögerungen in der Kommunikation mit GigE Vision Kameras kommen.

Persistente IP-Adresse festlegen

Wenn der PC die IP-Adresse des Netzwerkadapters nicht automatisch festlegen soll, sondern Sie manuell eine persistente IP-Adresse festlegen möchten, können Sie dies wie folgt in TwinCAT tun:

1. Öffnen Sie den **IP-Stack** des RT-Ethernet Adapters in der I/O-Konfiguration.
2. Wählen Sie den Tab **Parameter (Init)** und zeigen Sie mit einem Klick auf das **Plus (+)** vor `TcIoIpSettings` weitere Parameter an.
3. Legen Sie die Parameter `.IpAddress` und `.SubnetMask` fest.
4. Setzen Sie den Parameter `.ManualSettings` auf `TRUE`.



Dieselbe Einstellung wird automatisch vergeben, wenn Sie beim Anlegen eines Kamera-Objekts [▶ 39] eine manuelle IP-Adresse vergeben. Alternativ lässt sich eine persistente IP-Adresse auch über die Windows Netzwerkadapter-Eigenschaften festlegen.

HINWEIS

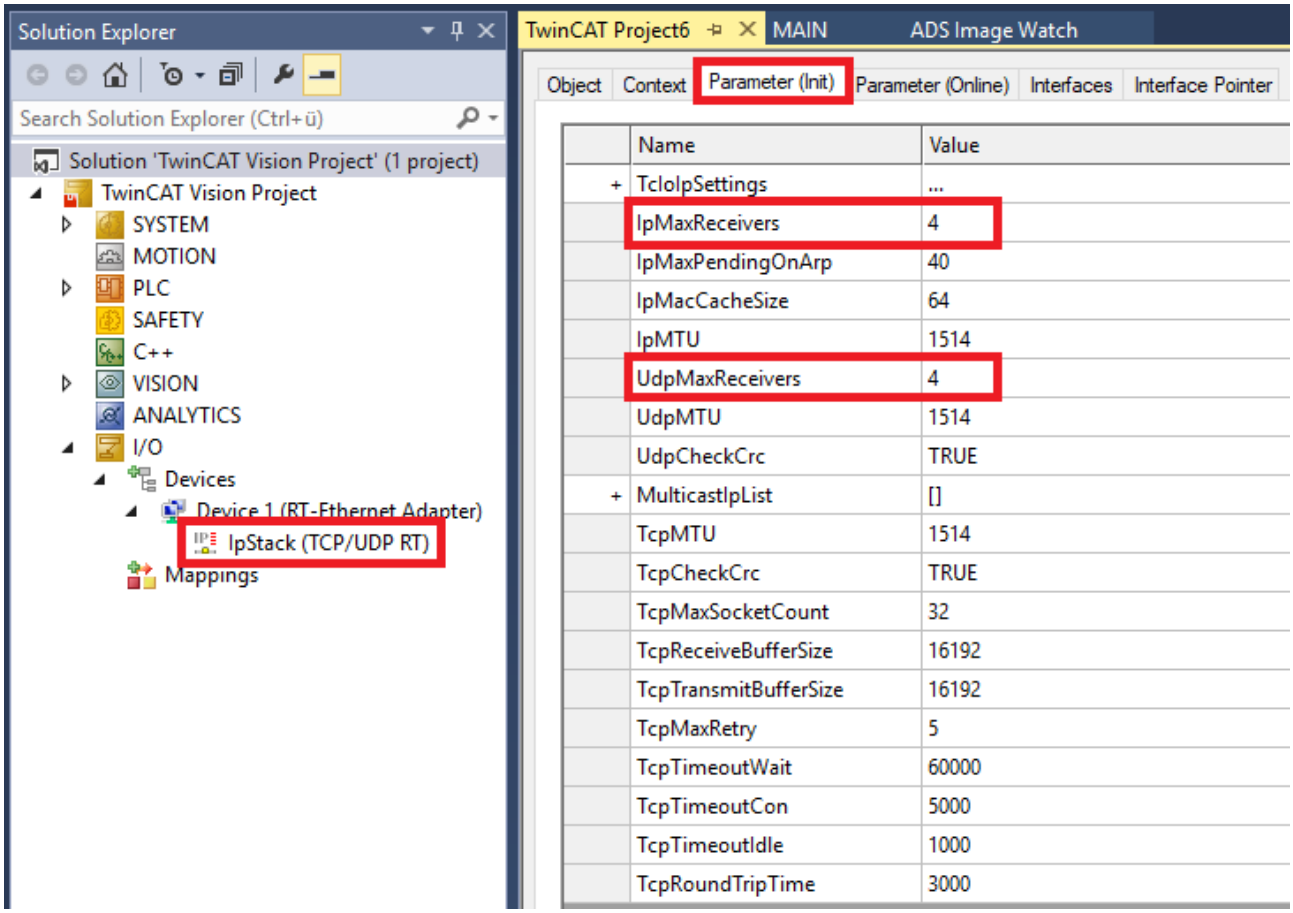
Persistente IP-Adresse

Um auszuschließen, dass sich die IP-Adresse des Netzwerk-Adapters ändern kann, legen Sie eine persistente IP-Adresse im IP-Stack des TwinCAT Projekts fest.

Betrieb von mehreren Kameras an einem Port

Wenn sie mehrere Kameras an demselben RT-Ethernet Adapter betreiben möchten, müssen Sie darauf achten, dass die UDP/IP-Einstellungen dies erlauben. Navigieren Sie dazu auf den UDP/IP-Stack unterhalb des RT-Ethernet-Geräts und wählen Sie den Karteireiter **Parameter (Init)**.

In der erscheinenden Parameter-Liste müssen nun `IpMaxReceivers` und `UdpMaxReceivers` so eingestellt werden, dass sie der doppelten Anzahl an Kameras entsprechen, die über diesen Adapter verbunden sind. Dies liegt daran, dass pro Kamera zwei Receiver existieren (GVCP- und GVSP-Modul). Die Standardeinstellung erlaubt zwei Kamera-Verbindungen, weswegen die Parameter auf `2 * 2 = 4` eingestellt sind.



Diagnose

Bei Problemen mit der Verbindung, speziell mit gesendeten oder empfangenen Frames, finden Sie im RT-Ethernet Adapter ein Tab mit Statistikinformationen und im IpStack das **Parameter (Online)** Tab mit einzelnen Counter-Informationen. Details finden Sie im Anhang/ Troubleshooting unter Kamerakommunikation [▶ 2964].

Promiscuous Mode

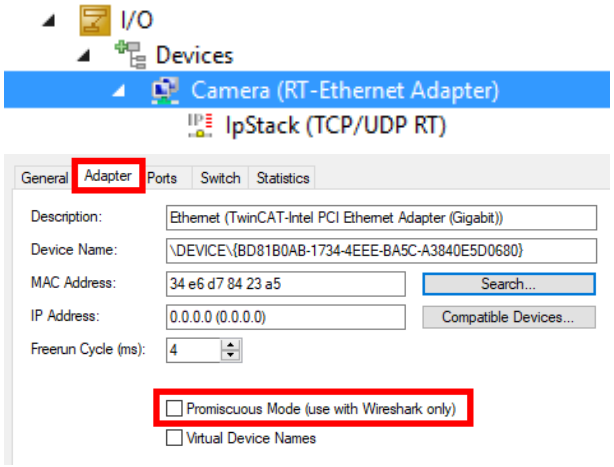
Der Promiscuous Mode ist eine Einstellung in TwinCAT RT-Ethernet Adaptern. Er wird nur für Debugging-Zwecke mit dem Tool Wireshark benötigt. Bei der Kommunikation mit GigE Vision-Geräten kann dieser Modus zu Problemen führen.

HINWEIS

Promiscuous Mode vermeiden

Bei der Kommunikation mit GigE Vision-Geräten kann dieser Modus zu Problemen führen. Deshalb betreiben Sie unbedingt GigE Vision-Geräte ohne Promiscuous Mode!

Dies können Sie überprüfen, indem Sie den entsprechenden RT-Ethernet Adapter in Visual Studio öffnen und auf den Reiter **Adapter** wechseln. Dort muss die Checkbox bei **Promiscuous Mode (use with Wireshark only)** deaktiviert sein.



5.1.5 CPU-Kerne und Tasks

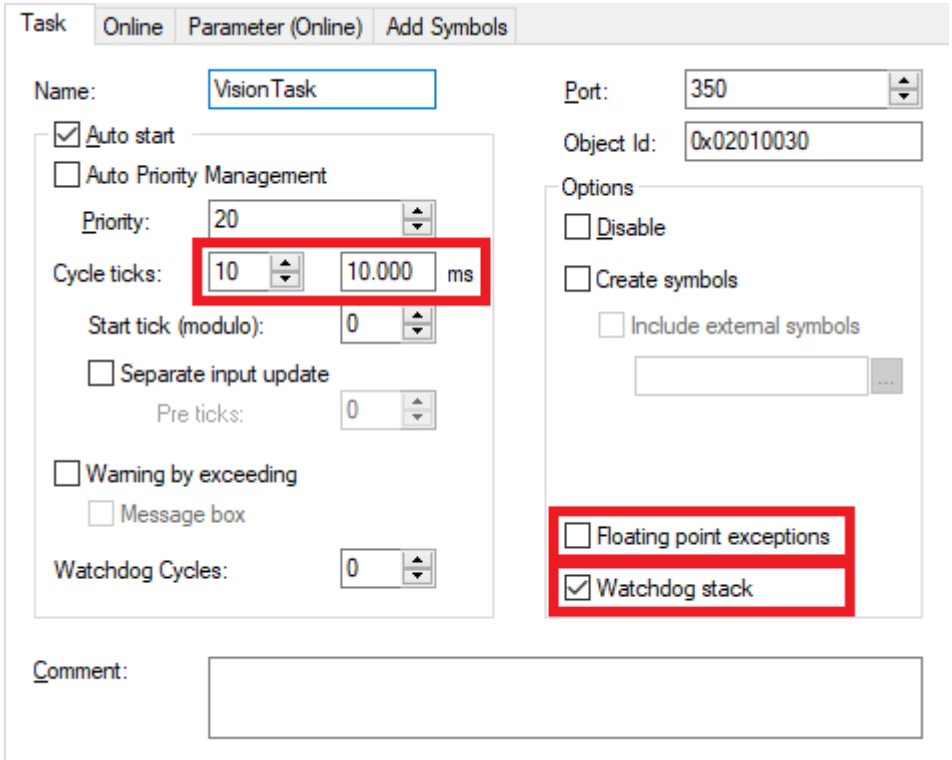
Die Konfiguration von Tasks und CPU-Kernen ist nicht spezifisch für TwinCAT Vision, sondern funktioniert für alle TwinCAT Komponenten gleich. Anleitungen dazu finden Sie sind unter Echtzeit-Einstellungen und Task-Einstellungen. Im Folgenden sind die Berührungspunkte und Erwägungen zu TwinCAT Vision beschrieben.

Eine TwinCAT Vision Applikation benötigt Tasks für verschiedene Zwecke:

- Zyklische Task zur Ausführung der SPS
- Zyklische Task zum Betrieb von Vision-Geräten
- Job Task für parallelisierte SPS-Ausführung
  (Optional; diese Art von Tasks ist separat im Kapitel Job Tasks [▶ 66] beschrieben.)

Konfigurieren von SPS-Tasks

Grundsätzlich werden alle für TwinCAT Vision benötigten zyklischen Tasks automatisch angelegt. Beim Anlegen einer SPS wird eine Task mit einer Zykluszeit von 10ms erstellt. Dies bedeutet, dass Bildverarbeitungsabläufe in der SPS alle 10ms wiederholt werden. Unter **SYSTEM > Tasks > [Taskname]** können Sie die Einstellungen der Task verändern.



Cycle ticks	Die Zyklus-Ticks multipliziert mit der Basiszeit des ausgewählten CPU-Kerns (standardmäßig 1ms) ergibt die oben beschriebene Zykluszeit der SPS.
Floating point exceptions	Diese Einstellung bestimmt, ob TwinCAT auf Fließkomma-Exceptions prüft oder nicht. Für Vision Applikationen wird empfohlen, diese Option zu deaktivieren. Zudem lassen diese Fehlermeldungen nicht auf den genauen Grund der Exception schließen.
Watchdog stack	Die Einstellung muss aktiviert sein, wenn in dem zugeordneten SPS-Programm Watchdogs [▶ 156] genutzt werden sollen.

⚠ WARNUNG

Floating point exceptions

Für SPS-Tasks, die im Wesentlichen Vision-Befehle ausführen, wird eindringlich empfohlen, die Option „Floating point exceptions“ zu deaktivieren. Anderenfalls können bei bestimmten TwinCAT Vision API-Funktionen unnötige Fehler auftreten, die zum Systemabbruch führen. Wenn jedoch auch Motion Control oder Safety Komponenten auf derselben SPS-Task ausgeführt werden, muss diese Option aktiv sein.

HINWEIS

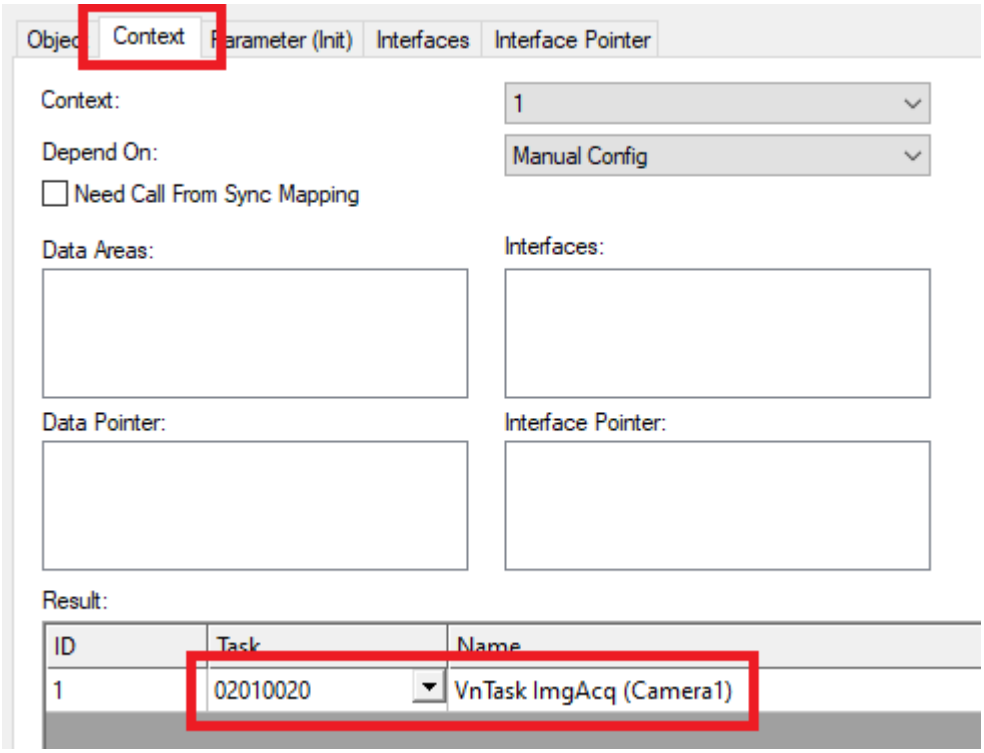
Floating point exceptions in Job Tasks

Wenn Sie zusätzlich Job Tasks [▶ 66] für Vision Applikationen verwenden und die „Floating point exceptions“ in der SPS Task deaktiviert haben, müssen Sie diese Option ebenfalls in den zugeordneten Job Tasks deaktivieren. Ansonsten können weiterhin Exceptions und Systemabbrüche auftreten.

Konfigurieren von Geräte-Tasks

Beim Anlegen von Vision-Geräten werden benötigte Tasks automatisch generiert. Die Zykluszeiten dieser Tasks können wie zuvor beschrieben angepasst werden. Für die meisten Fälle wird empfohlen, die Zykluszeiten dieser Tasks nicht zu verändern. Dies ist lediglich in Sonderfällen erforderlich.

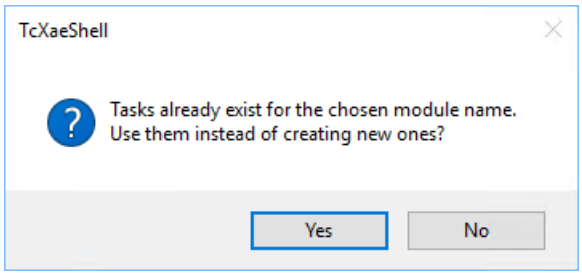
Falls die Verknüpfung eines Vision-Geräts mit einer Task doch einmal angepasst werden muss, geschieht dies im Karteireiter **Context** der jeweiligen TcCOM-Objekte eines GigE Vision Camera [▶ 132] oder File Source [▶ 139] Objekts:



Folgende TcCOM-Objekte müssen mit einer zyklischen Task verknüpft sein, sofern sie benutzt werden:

- Camera Image Acquisition [▶ 133]
- Camera Image Acquisition Simulation [▶ 133]
- ADS Communicator [▶ 133]
- File Image Acquisition [▶ 139]

Wenn beim Erstellen eines Vision-Geräts schon passende Tasks unter dem gewählten Gerätenamen existieren, können diese optional wiederverwendet werden. In diesem Fall erscheint folgendes Fenster, klicken Sie auf **Ja** um die Wiederverwendung zu bestätigen:



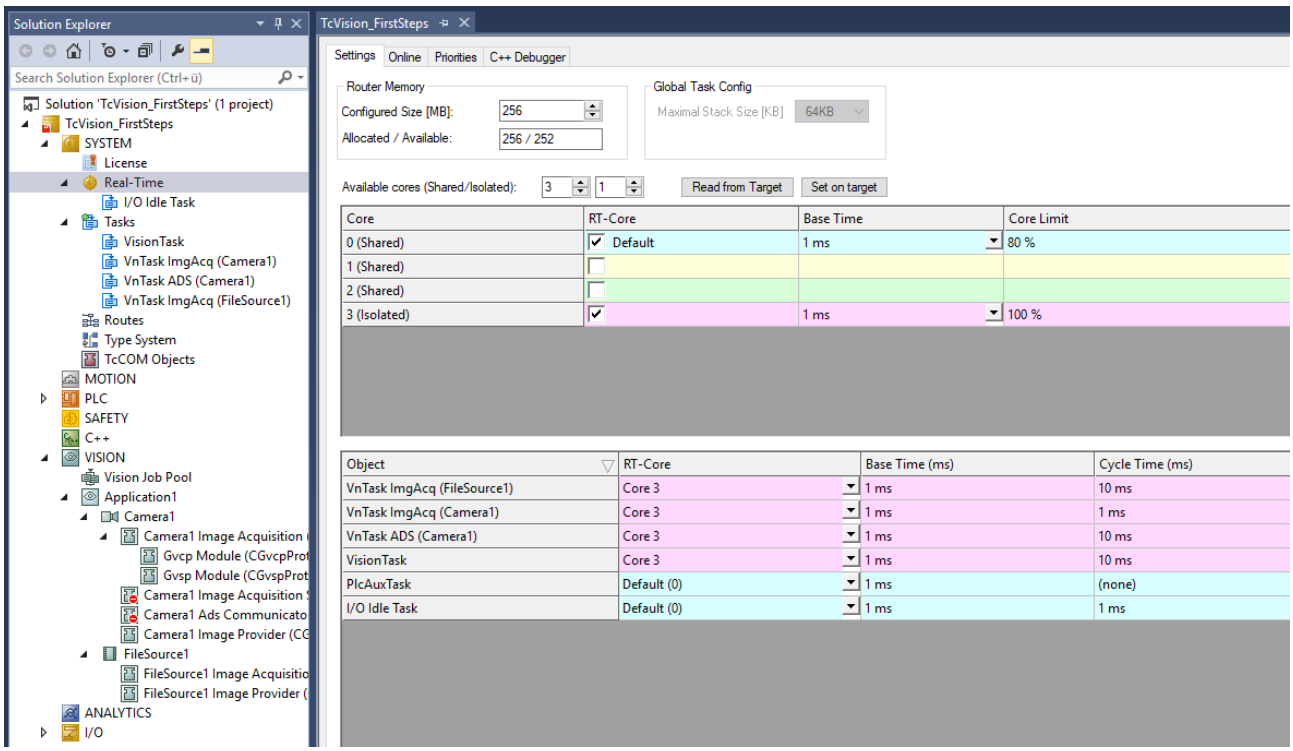
Wiederwendung von Geräte-Tasks

Bei der Wiederwendung von Geräte-Tasks sollte sichergestellt werden, dass diese nicht von mehreren Vision-Geräten gleichzeitig genutzt werden.

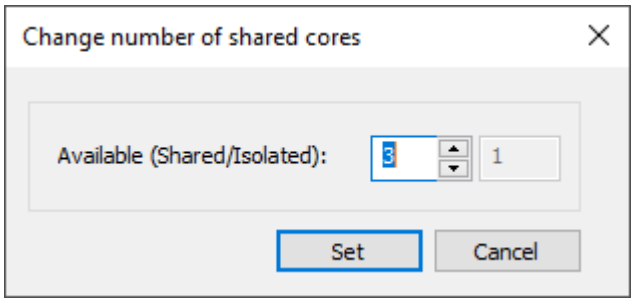
Zuordnung zu CPU-Kernen

Die Zuordnung von Tasks zu CPU-Kernen erfolgt im TwinCAT Projekt unter **SYSTEM > Real-Time > Settings**. Bei Multicore-Systemen ermöglicht TwinCAT die Unterscheidung zwischen Windows-Kernen und isolierten Kernen. Bei Windows-Kernen teilen sich das Betriebssystem und die TwinCAT-Anwendung die Prozessorzeit. Der Echtzeitanteil kann dabei auf 10% bis 90% begrenzt werden. Im Gegenzug stehen die isolierten Kerne vollständig der TwinCAT-Anwendung zur Verfügung. Aus diesem Grund wird die Verwendung von isolierten Kernen für Vision-Anwendungen empfohlen.

Um die aktuelle CPU-Kern-Konfiguration des Zielsystems anzuzeigen, klicken Sie auf **Read from Target**.



Um die CPU-Kern-Konfiguration des Zielsystems zu ändern, klicken Sie auf **Set on Target** und wählen die gewünschte Verteilung von geteilten und isolierten Kernen aus.



Hiernach ist ein Neustart des Zielsystems erforderlich.

Empfehlung

Wir empfehlen, für die Vision-Tasks (Geräte und SPS) mindestens einen dezidierten, isolierten Kern zu benutzen. Falls der TcVnService durch das Abspeichern oder Laden von Bildern (durch Kamera Streams, ein File Source Control oder direkt aus der SPS) stark beansprucht wird, empfehlen wir einen Kern komplett für Windows zu reservieren oder den Echtzeitanteil eines Kerns stark zu begrenzen.

Wenn mehrere Tasks einem Kern zugeordnet sind, muss die Priorität beachtet werden. In der Regel sollte den Tasks mit kürzerer Zykluszeit eine höhere Priorität zugewiesen werden.

5.1.6 Job Tasks

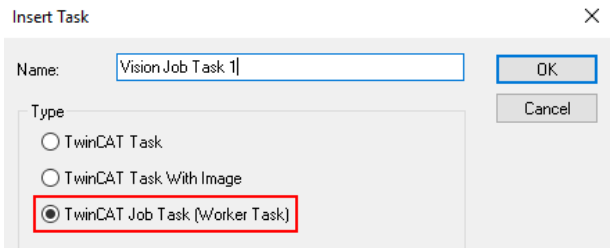
Job Tasks werden von TwinCAT Vision zur Parallelisierung von Funktionen verwendet. Abhängig vom verwendeten System, dessen Konfiguration, den verwendeten API-Funktionen, dem Bildinhalt sowie der Anzahl an Job Tasks, kann dies zu einer Reduzierung der Ausführungszeit führen. Die unterstützenden Funktionen sind durch den Kommentar “Can use available TwinCAT Job Tasks for executing parallel code regions.” in der API-Referenz [▶ 146] gekennzeichnet. Über den Vision Job Pool [▶ 140] erfolgt die Zuweisung, welche Job Tasks von TwinCAT Vision verwendet werden. Job Tasks müssen nur einmalig angelegt und konfiguriert werden; danach geschieht die Parallelisierung automatisch in der Runtime.

Nutzen von Job Tasks

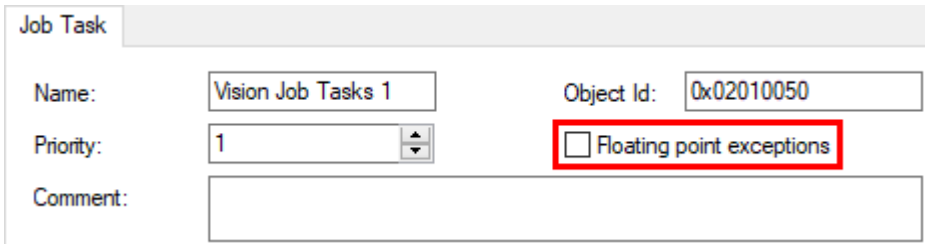
Der tatsächliche Nutzen von Job Tasks muss im Einzelfall geprüft werden. Dieser ist abhängig vom verwendeten System, dessen Konfiguration, den verwendeten API-Funktionen, dem Bildformat, dem Bildinhalt sowie der Anzahl an Job Tasks.

Verwendung

Job Tasks können - wie andere Tasks auch - über **Solution Explorer > SYSTEM >** Rechtklick auf **Tasks > Add new item...** erstellt werden. Im Nachfolgenden **Insert Task Dialog** wählen Sie dann den Typ **TwinCAT Job Task (Worker Task)** aus.



Danach kann - wie bei allen anderen Tasks auch - eine entsprechende Kernzuweisung über **Solution Explorer > SYSTEM > Real-Time** erfolgen. Beachten Sie hierbei die Hinweise unter Einschränkungen und Empfehlungen [▶ 68].



HINWEIS

Floating point exceptions

Diese Einstellung bestimmt, ob TwinCAT auf Fließkomma-Exceptions prüft oder nicht. Hier sollte die gleiche Option gewählt werden, wie in der Konfiguration der entsprechenden SPS Tasks [▶ 63]. Ansonsten können Exceptions und Systemabbrüche auftreten.

Die Zuordnung, welche Job Tasks von TwinCAT Vision verwendet werden, erfolgt über die Parameterliste des Vision Job Pool [▶ 140]s. Der Vision Job Pool ist im Solution Explorer unterhalb des Vision Knotens zu finden, sobald dort eine Applikation angelegt wurde.

Priorisierung

Priorisierung des Vision Job Pool Zugriffs

Wenn mehrere parallel ausgeführte Programme sich den Vision Job Pool teilen, bekommt generell immer das Programm mit höchster Task-Priorität Zugriff auf die Job Tasks. Diese Zuordnung ist jedoch dynamisch, d. h., wenn das Programm mit höchster Priorität sich aktuell in einem sequenziellen Bearbeitungsabschnitt befindet, können niedriger priorisierte Programme die Job Tasks nutzen. Sobald ein höher priorisiertes Programm dann Aufgaben parallel bearbeiten kann, werden die Aufgaben von niedriger priorisierten Programmen verdrängt. Die Master Task des niedriger priorisierten Programmes ist aber in der Lage, eine parallelisiert begonnene Aufgabe allein zu beenden, sodass nicht auf eine erneute Verfügbarkeit von Job Tasks gewartet werden muss.

Mehrere Tasks auf einem Kern
Wie den Einschränkungen und Empfehlungen zu entnehmen ist, sollte sich eine Job Task immer allein auf einem Kern befinden. Andernfalls gelten die Task-Prioritäten.

Einschränkungen

- Die Job Tasks sind aktuell nur für TwinCAT Vision zu verwenden.
- Es dürfen nicht mehrere Job Tasks demselben Kern zugeordnet werden.
- Eine Job Task darf nicht demselben Kern zugeordnet werden wie eine Vision PLC Task.
- Eine Job Task darf nicht demselben Kern zugeordnet werden wie die PlcAuxTask.

Empfehlungen

- Jede Job Task sollte jeweils einem eigenen, isolierten Kern zugeordnet werden.
- Sollte doch eine weitere zyklische Task auf demselben Kern laufen wie eine Job Task, sollte die Priorität der zyklischen Task höher sein als die der Job Task. Zudem sollte beachtet werden, dass dadurch die Parallelisierung eingeschränkt wird und im schlimmsten Fall nicht zur Verfügung steht.
- Keine Job Task sollte dem „Default“ Kern zugewiesen werden.

Wenn gleichzeitig zyklische TwinCAT Tasks oder auch das Windows Betriebssystem auf dem Kern Rechenzeit beanspruchen, wird eine Analyse durch die unterschiedliche Belastung der Job Tasks erschwert.

5.2 VISION Knoten

Die Konfiguration von bildverarbeitungsrelevanten Komponenten erfolgt über den VISION Knoten und dessen Sub-Elemente. Er enthält die Karteireiter Service Configuration [▶ 68] und Logging [▶ 71].

Anzeigen des VISION Knotens

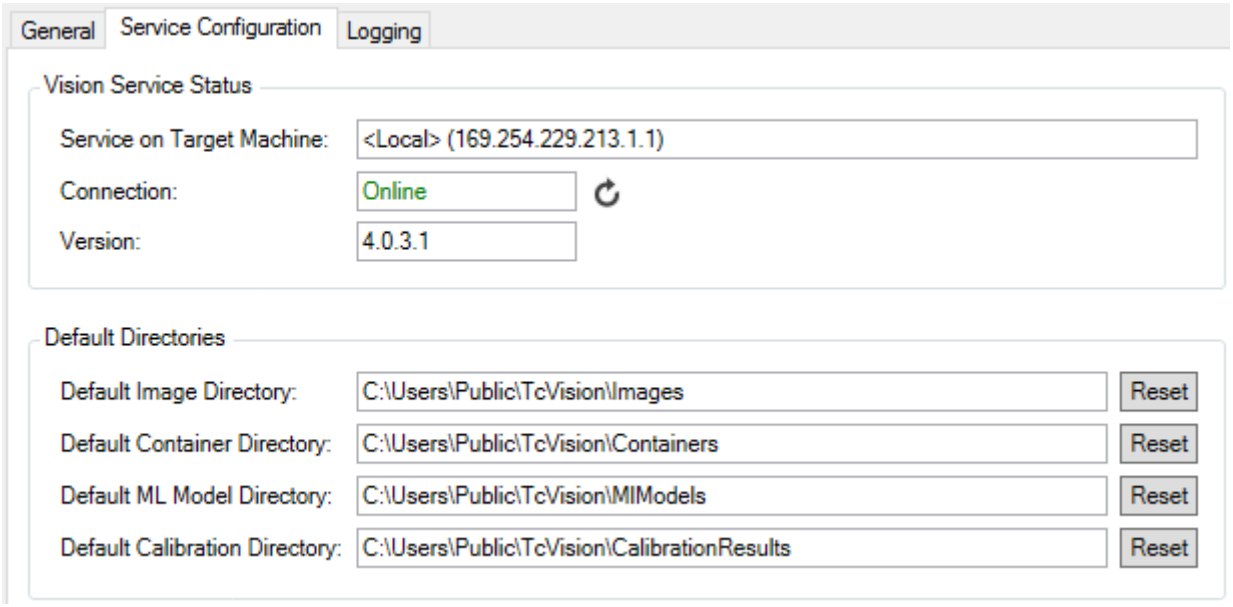
Falls der VISION Knoten in ihrem TwinCAT-Projekt nicht angezeigt wird, tätigen Sie einen Rechtsklick auf Ihr TwinCAT-Projekt und wählen Sie **Show Hidden Configurations > Show VISION Configuration** um den Vision Knoten einzublenden. Der nächste Schritt ist die Erstellung eines Application Knotens [▶ 71].

5.2.1 Service Configuration

Der Vision-Service bietet ein Nicht-Echtzeit-Gegenstück für einige Echtzeit-Komponenten (z.B. Funktionsblöcke). Dazu implementiert der Vision-Service einen ADS-Server, was den Zugriff sowohl aus der Echtzeit, als auch aus der Entwicklungsumgebung ermöglicht.

Der Vision-Service ermöglicht den TwinCAT Vision API-Funktionsblöcken den indirekten Zugriff auf die Festplatte aus der Echtzeitumgebung, z.B. um Bilder zu laden oder zu speichern.

Die Konfiguration des Vision-Services erfolgt unter **VISION > Service Configuration**.



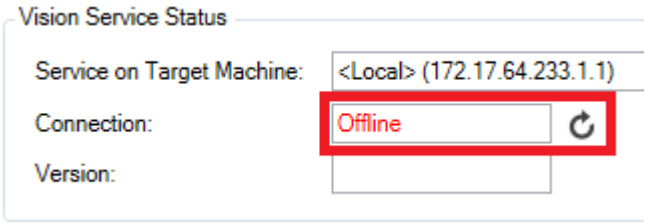
Vision Service Status

Service on Target Machine	Zielsystem, auf dem der TwinCAT Vision Service läuft.
Connection	Verbindungsstatus zwischen Entwicklungsumgebung und Service. Dies ist ein guter Indikator, ob der Service ordnungsgerecht läuft.
Version	Versionsnummer des TwinCAT Vision Service.

Vor der ersten Verwendung muss der Vision-Service auf dem Zielsystem installiert werden, siehe Installationsanleitung [▶ 18].

Manuelles Starten des Vision-Service

Normalerweise wird der Vision Service automatisch zusammen mit TwinCAT gestartet. Falls der Vision Service dennoch offline ist, kann er mit einem TwinCAT Neustart (Klick auf oder) reaktiviert werden. Falls dies nicht funktioniert oder der Service wiederholt nicht startet, sollte er bei TwinCAT neu registriert werden, siehe dazu: Vision Service startet nicht [▶ 2951].



Default Directories

Diese Pfade legen fest, an welcher Stelle auf dem Zielsystem TwinCAT Vision standardmäßig nach Dateien suchen und diese Ablegen soll.

Default Image Directory	Dieser Pfad wird standardmäßig von den Funktionsblöcken zum Lesen und Schreiben von Bildern verwendet: FB_VN_ReadImage [▶ 1648] FB_VN_WriteImage [▶ 1656]
Default Container Directory	Dieser Pfad wird standardmäßig von den Funktionsblöcken zum Lesen und Schreiben von Containern verwendet: FB_VN_ReadContainer [▶ 1646] FB_VN_WriteContainer [▶ 1654]
Default ML-Model Directory	Dieser Pfad wird standardmäßig von den Funktionsblöcken zum Lesen und Schreiben von ML-Modellen verwendet: FB_VN_ReadNeuralNetwork [▶ 1660] FB_VN_ReadMlModel [▶ 1650] FB_VN_WriteMlModel [▶ 1658]
Default Calibration Directory	Dieser Pfad wird standardmäßig von den Funktionsblöcken zum Lesen und Schreiben von Kalibrationsergebnissen verwendet: FB_VN_ReadCalibrationResult [▶ 1644] FB_VN_WriteCalibrationResult [▶ 1652]

Eine vollständige Liste aller wichtigen Pfade finden Sie in der Sektion Wichtige Pfade [▶ 2947].

Vision Service Log

Alle wichtigen Service-Ereignisse werden auf dem Zielsystem in die Logdatei TcVnService.log geschrieben, welche auf dem Zielsystem unter *C:\ProgramData\Beckhoff\TcVnService* zu finden ist. Bei Supportanfragen zum TwinCAT Vision Service ist die Logdatei, sowie der Zeitpunkt, zu dem das Verhalten aufgetreten ist, mitzusenden. Auf dem Entwicklungssystem gibt es zudem einen Logger [▶ 71] für alle Komponenten unterhalb des Vision-Knotens.

Vision Service Cache Einstellungen

Per Default hinterlegt der Vision Service die eingelesenen Bilder im Cache, um die angeforderten Bilder beim nächsten Abruf schneller bereitstellen zu können. Diese optimierte Betriebsweise deckt die Anforderungen der meisten Anwendungen ab und wird daher empfohlen.

Für den Fall, dass sich der Bildinhalt ändert, der Name der Datei jedoch gleichbleibt, kann das Cachen der Bilder abgeschaltet werden. Öffnen Sie dazu die Einstellungen des Vision Service unter *C:\ProgramData\Beckhoff\TcVnService\settings.json* auf dem Target System, ändern Sie den Default Wert des Parameters updateCacheOnReadSize von 0 auf 1 und speichern die Datei.

Vision Service beenden

Zum Ändern der Einstellungen muss zuerst der TwinCAT Vision Service (TcVnService.exe) über den Taskmanager beendet werden, da ansonsten die Änderungen nicht übernommen werden. Danach kann der Service durch ein Restart in Config Mode wieder gestartet werden.

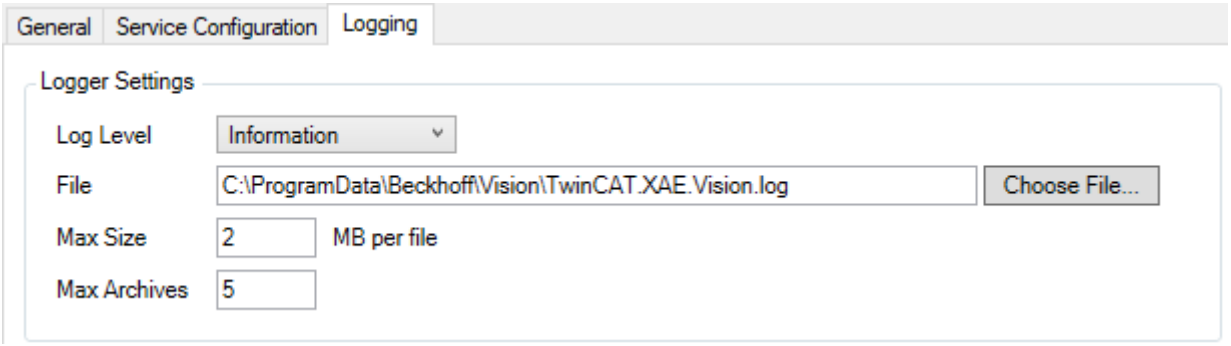
HINWEIS

Inkonsistente Daten

Stellen Sie sicher, dass kein gleichzeitiger Zugriff von mehreren Prozessen auf dieselbe Datei erfolgt und es so zu einer Dateikorruption oder inkonsistenten Daten kommen kann.

5.2.2 Logging

Der Vision-Knoten besitzt einen Logger, welcher abhängig vom Log Level alle wichtigen Ereignisse der TwinCAT Vision Komponenten unterhalb des VISION Knotens protokolliert. Die Einstellungen dazu sind unter **VISION > Logging** zu finden.



Dabei können folgende Einstellungen vorgenommen werden:

Log Level

Level	Beschreibung
Error	Nur Fehlermeldungen werden geloggt.
Warning	Fehlermeldungen und Warnungen werden geloggt.
Information	Fehlermeldungen, Warnungen und wichtige Ereignisse werden geloggt. [default]
Verbose	Fehlermeldungen, Warnungen und alle wesentlichen Ereignisse werden geloggt.

Log-Datei

Standardmäßig wird die Logdatei mit dem Namen *TwinCAT.XAE.Vision.log* in dem Verzeichnis *C:\ProgramData\Beckhoff\Vision* gespeichert. Über den Karteireiter **Logging** kann dies aber auch geändert werden. Genauso wie die maximale Größe einer Logdatei und die maximale Anzahl an Dateien. Sind die Limits erreicht, so werden die alten Daten überschrieben.

Bei Supportfragen zu zuvor aufgetretenem Verhalten innerhalb des Vision Knotens ist die Logdatei, sowie der Zeitpunkt, zu dem das Verhalten aufgetreten ist, mitzusenden.

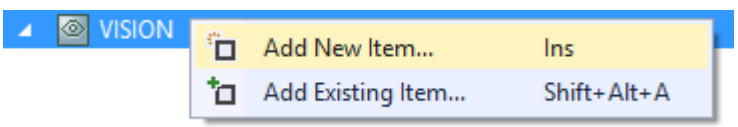
Weitere Logger

Einen weiteren Logger hat der Vision Service [▶ 68] auf dem Zielsystem. Zudem gibt es eine Log-Datei für die TwinCAT Vision Installation. Alle relevanten Logging-Pfade sind in der Sektion Wichtige Pfade [▶ 2947] aufgeführt.

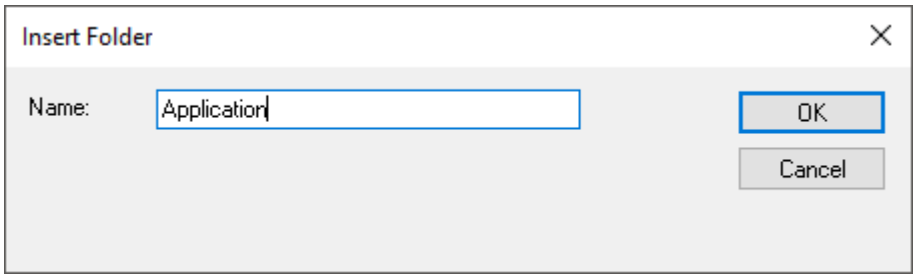
5.3 Application Knoten

Eine TwinCAT Vision Konfiguration kann mehrere Applikationsknoten enthalten, die alle innerhalb des VISION Knotens angesiedelt sind. Ein Applikationsknoten wird wie folgt erstellt:

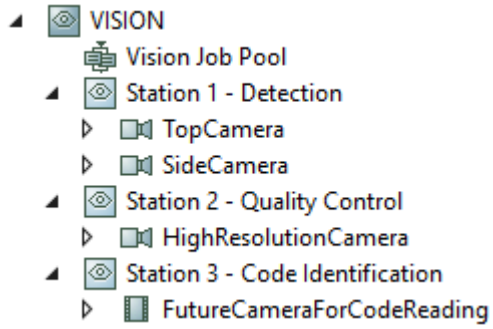
1. Rechtsklick auf den **VISION Knoten** → **Add New Item…**



2. Tragen Sie einen Namen für den Applikationsknoten ein und klicken Sie **OK**.



Unter dem Applikationsknoten werden GigE Vision Camera Knoten [▶ 72] und File Source Knoten [▶ 135] angelegt. Der Sinn von Applikationsknoten ist somit das Kapseln von thematisch zusammenhängenden Vision-Geräten. Das folgende Bild zeigt eine Beispiel-Struktur mit mehreren Vision-Applikationen:



Zudem ermöglichen Applikationsknoten das gemeinsame Exportieren/Importieren [▶ 144] von mehreren Vision-Geräte-Konfigurationen in einer einzigen XTI-Datei.

5.4 GigE Vision Camera-Objekte

Jedes GigE Vision Camera-Objekt repräsentiert jeweils eine Kamera innerhalb eines Projektes. Ein Projekt kann mehrere Kameraobjekte beinhalten.

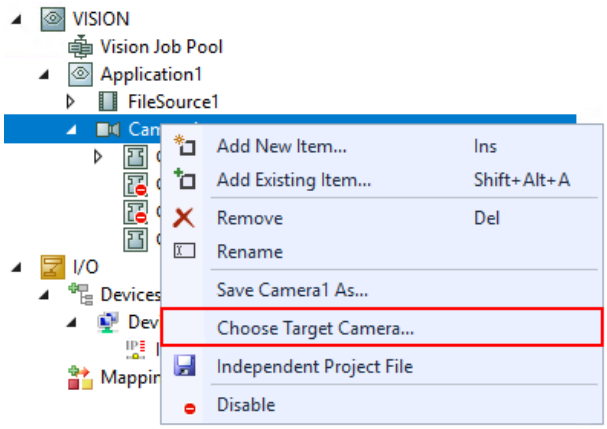
Anlegen eines GigE Vision Camera-Objekts

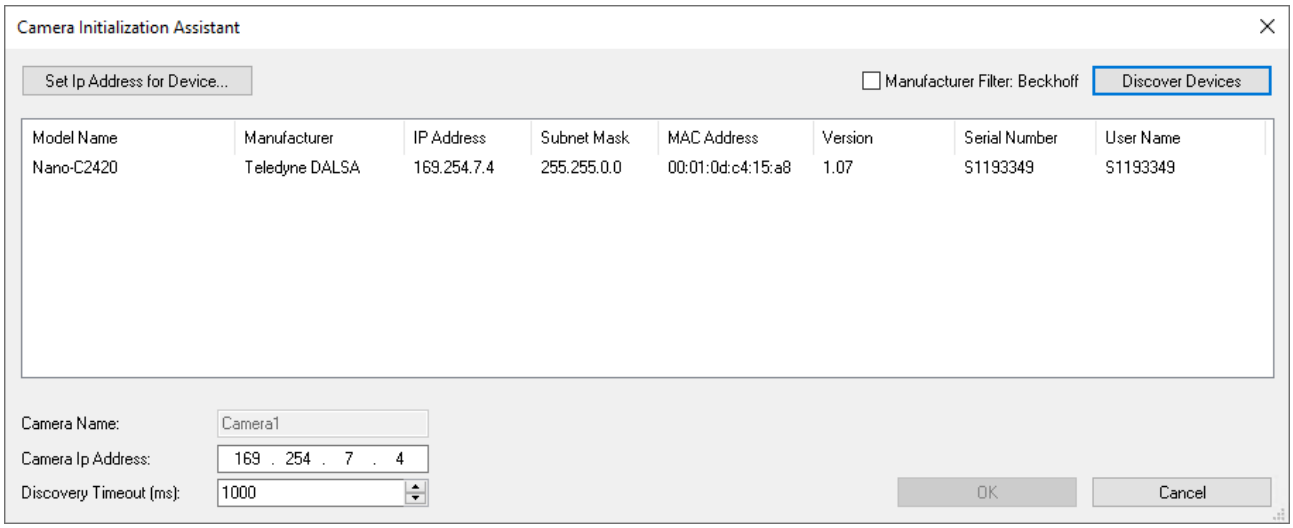
Das Anlegen einer Kamera ist in den First Steps [▶ 28] unter GigE Vision Kamera anlegen [▶ 39] beschrieben. Beachten Sie dabei die Hinweise zur Konfiguration von Netzwerkadaptern [▶ 58] und Geräte-Tasks [▶ 63].

Nach dem Anlegen einer Kamera wird automatisch versucht, die GenAPI-Datei von der Kamera zu lesen. Darin ist die vollständige Beschreibung des Parameterbaums der Kamera hinterlegt (siehe GigE Vision Standard [▶ 131]). Auf Basis dieser Beschreibung wird daraufhin der Konfigurationsbaum im Configuration Assistant [▶ 79] bereitgestellt und die aktuellen Parameterwerte werden von der Kamera ausgelesen.

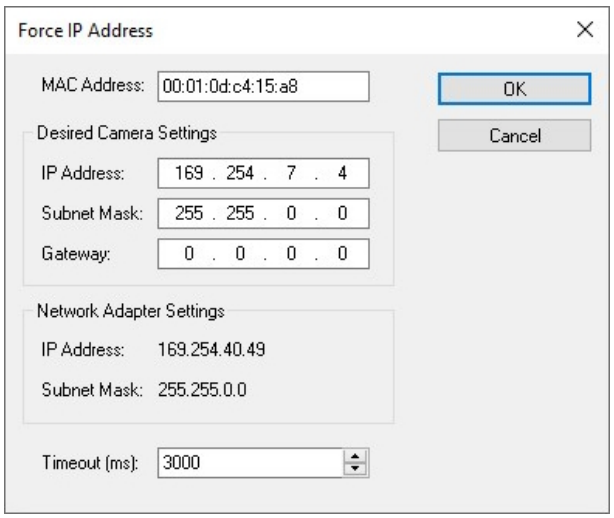
Auswahl einer verbundenen Kamera

Um eine mit dem Zielsystem verbundene Kamera mit einem GigE Vision Camera-Objekt zu verknüpfen, kann über einen Rechtsklick auf dem entsprechenden Kameraobjekt und Auswahl von **Choose Target Camera** der **Camera Initialization Assistant** geöffnet werden. Alternativ kann dieser Dialog auch über die Connection Optionen auf dem General [▶ 74] Tab geöffnet werden.





Camera Name	Anzeige des Namens des ausgewählten Kameraobjektes.
Camera Ip Address	IP-Adresse die in den TcCOM Parametern gespeichert ist. Der Eintrag wird bei Auswahl einer Kamera überschrieben und mit **OK** auch im Modul aktualisiert.
Discovery Timeout	Das Discovery Timeout beschreibt die Zeit, in der auf Antwort von Kameras gewartet wird, die an dem entsprechenden Netzwerkport angeschlossenen wurden. Einheit: Millisekunden (ms) Standard: 1000
Set Ip Address for Device	Öffnet den folgenden Force Ip Adress Dialog, um die IP-Adresse und die Subnetzmaske einer Kamera zu setzen. Die festgelegte IP-Adresse und Subnetzmaske wird in dem TcCOM-Parameter `CameraIPAddress` im Image Acquisition Objekt [▶ 106] gespeichert.
Manufacturer Filter: Beckhoff	Aktiviert einen Filter, so dass in der Ergebnistabelle nur noch Kameras des Herstellers Beckhoff angezeigt werden.
Discover Devices	Mit **Discover Devices** wird der Suchvorgang gestartet.
Ergebnistabelle der gefundenen Kameraverbindungen	Auflistung der gefundenen Kameras und Anzeige der folgenden ausgelesenen Kamerainformationen: Modelname, Hersteller, IP-Adresse, Subnetzmaske, MAC-Adresse, Version, Seriennummer und Benutzerkennung



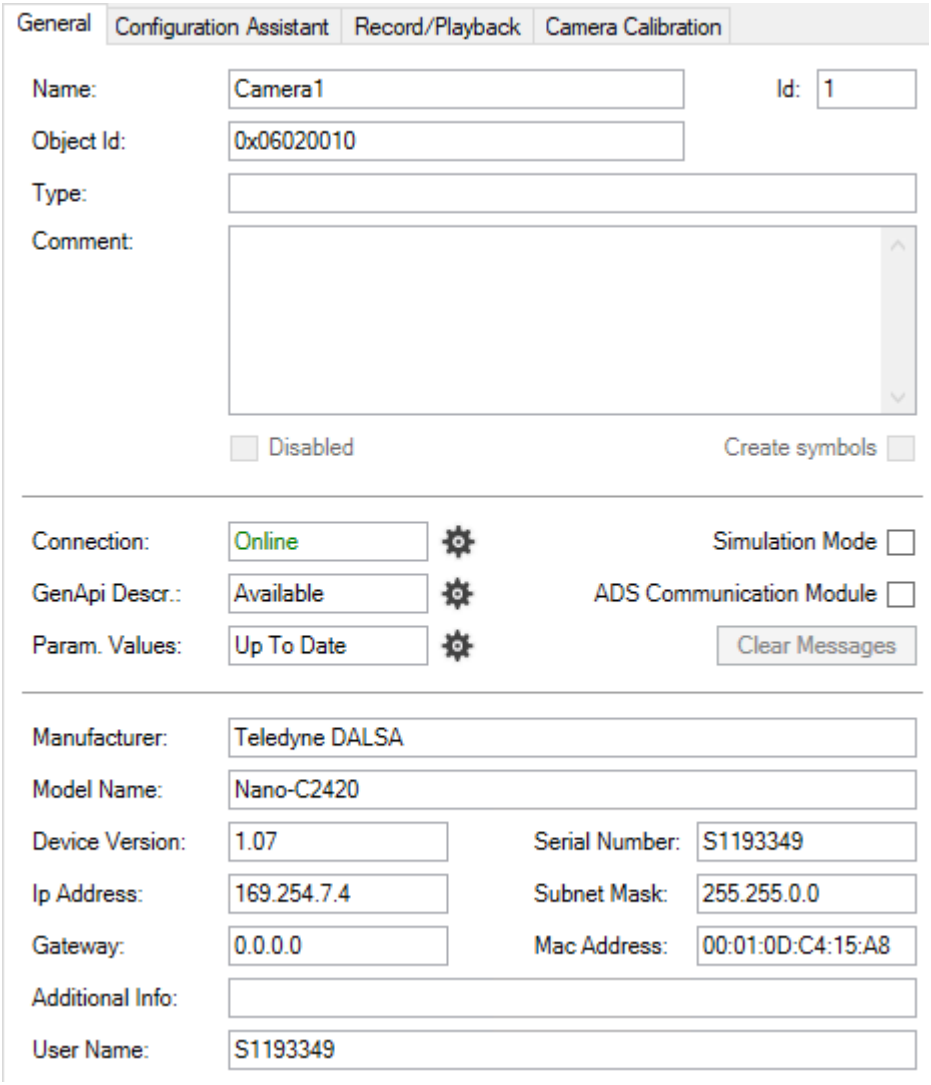
MAC Address	MAC-Adresse der ausgewählten Kamera.
Desired Camera Settings	Eingabe der gewünschten IP-Adresse, Subnetzmaske und Gateway auf die die Kamera gesetzt werden soll.
Network Adapter Settings	Aktuelle IP-Adresse und Subnetzmaske des Netzwerkadapters der mit der Kamera verbunden ist.
Timeout	Timeout für die Antwort der Kamera auf das Kommando zum Forcieren der IP-Einstellungen.

Exportieren/Importieren von GigE Vision Camera-Objekten

Wenn Sie die Konfiguration einer Kamera abspeichern oder in ein Projekt importieren möchten, folgen Sie der Anleitung im Kapitel Exportieren/Importieren von Konfigurationen [▶ 144].

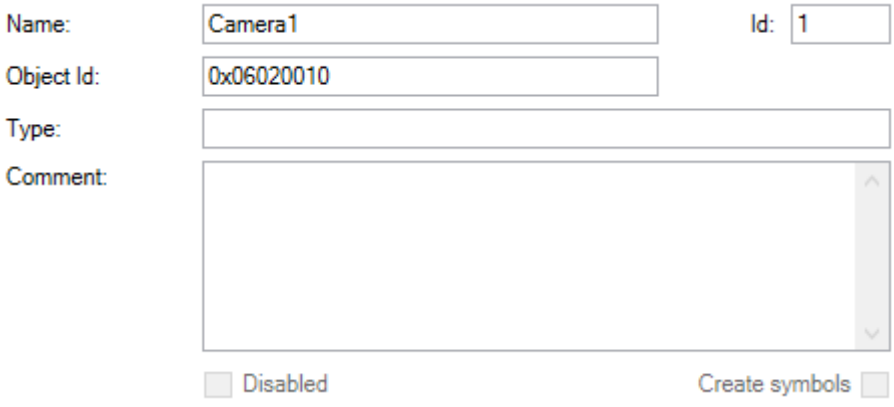
5.4.1 General

Der Karteireiter General gibt einen Überblick zum Kamera-Objekt in TwinCAT und zur angeschlossenen Kamera.



Allgemeine Objekt-Informationen

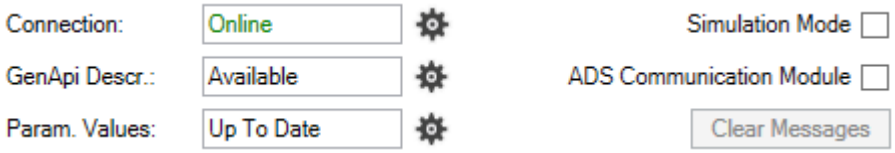
Im oberen Teil sind die allgemeinen Informationen des Kamera-Objekts angegeben:



Name	Gewählter Name des Objekts. **Vergeben Sie keinen Objektnamen doppelt!**
Object Id	Automatisch vergebene, innerhalb eines Projektes eindeutige 32-Bit-Identifikationsnummer.
Type	Objekt-Typ.
Comment	Frei editierbares Feld für eigene Notizen.
Id	Identifikationsnummer der Kameramodule, welche fortlaufend vergeben wird. Wird ein Kameramodul gelöscht, wird die Nummer wieder frei und kann einem neuen Kameramodul zugewiesen werden.

Status der Kamera-Instanz

Im mittleren Teil befinden sich Status-Informationen zu der Kamera-Instanz:



Die folgende Tabelle erklärt die einzelnen Elemente:

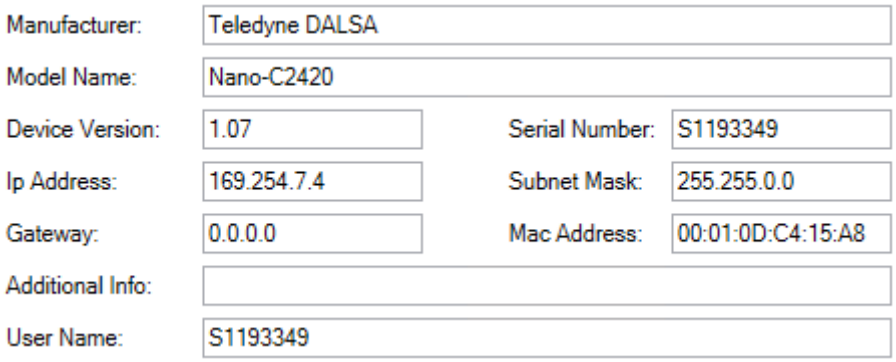
Connection	Hier wird der Verbindungsstatus zur Kamera angezeigt. Über den Button kann mit **Refresh Camera Status** die Anzeige aktualisiert werden. Zusätzlich besteht die Möglichkeit mit **Choose Target Camera** den Dialog zur Auswahl einer Kamera zu öffnen Möglicher Status: • **Unknown**: Die Verbindung zur Kamera kann nicht überprüft werden. Das Kameramodul könnte sich nicht im OP-Zustand befinden oder das Kameramodul wurde unvollständig angelegt. • **Online**: Die Verbindung zur Kamera steht. • **Offline**: Die Verbindung zur Kamera ist nicht vorhanden. • **Disabled**: Der Kameraknoten ist deaktiviert. Es wird nicht mit der Kamera kommuniziert. • **Not in Config Mode**: Der Verbindungsstatus kann nur im TwinCAT Config Mode an dieser Stelle wiedergegeben werden. • **Loading**: Das Kameramodul wird initialisiert.
GenApi Descr.	Hier wird angegeben, ob die GenApi Struktur (Feature-Baum im Configuration Assistant [▶ 79]) erfolgreich erstellt wurde. Mögliche Status: • **Available**: GenApi Struktur wurde erfolgreich erstellt. • **Not Available**: GenApi Struktur wurde nicht erstellt. Es können keine Features angezeigt werden. Über den Button sind zudem folgende Aktionen möglich: • **Read GenApi Description from Camera**: Die GenApi Beschreibung der Kamera wird eingelesen. Dabei werden alle Features im Konfigurationsbaum [▶ 83] zurückgesetzt. • **Export GenApi Description**: Die GenApi Beschreibung der Kamera wird lokal gespeichert.

Param. Values	Mögliche Status: • **Up To Date**: Alle Features der Kamera haben Werte und sind aktuell (Kamera Online). • **Complete**: Alle Features der Kamera haben Werte, aber könnten veraltet sein (Kamera Offline). • **Incomplete**: Nicht alle Features der Kamera haben Werte. Über den Button sind zudem folgende Aktionen möglich: • **Write Local Changes:** Alle lokalen Parameterwerte, die von den Kamerawerten abweichen (orange im Konfigurationsbaum [▶ 83] gekennzeichnet), werden auf die Kamera geschrieben. • **Revert Local Changes**: Alle Änderungen im Feature-Baum, welche noch nicht auf die Kamera geschrieben wurden, werden zurückgesetzt.
Simulation Mode	Im Simulationsmodus können Bilder aus aufgenommenen Kamera-Streams in die SPS gesendet werden, anstatt dass eine echte Kamera Bilder liefert (siehe Record/Playback [▶ 109]). Wenn der Simulationsmodus aktiviert ist, wird das Image Acquisition Objekt automatisch deaktiviert und das Image Acquisition Simulation Objekt aktiviert. Wenn der Simulationsmodus deaktiviert ist, wird das Image Acquisition Objekt automatisch aktiviert und das Image Acquisition Simulation Objekt deaktiviert. Die entsprechenden Objekt-Verknüpfungen (siehe Kamera-Details [▶ 132]) werden jeweils automatisch angepasst. Standard: deaktiviert.
ADS Communication Module	Wird für das Aufzeichnen eines Kamera-Streams benötigt, siehe Record/Playback [▶ 109]. Diese Checkbox aktiviert bzw. deaktiviert lediglich das Ads Communicator Objekt des Kamera-Objekts. **Beachten Sie, dass bei aktiviertem Ads Communicator Objekt von der Kamera empfangene Bilder für kurze Zeit noch intern referenziert sind, sodass sie nicht in anzeigbare Bilder transformiert werden können.** Für weitere Details siehe Anzeigbare Bilder [▶ 161]. Standard: deaktiviert.

Clear Messages	Mit diesem Button können Sie die Meldungen löschen, die von diesem Kameraobjekt geschrieben wurden. Der Button ist nur aktiv wenn Meldungen vorhanden sind.

Informationen zur Kamera

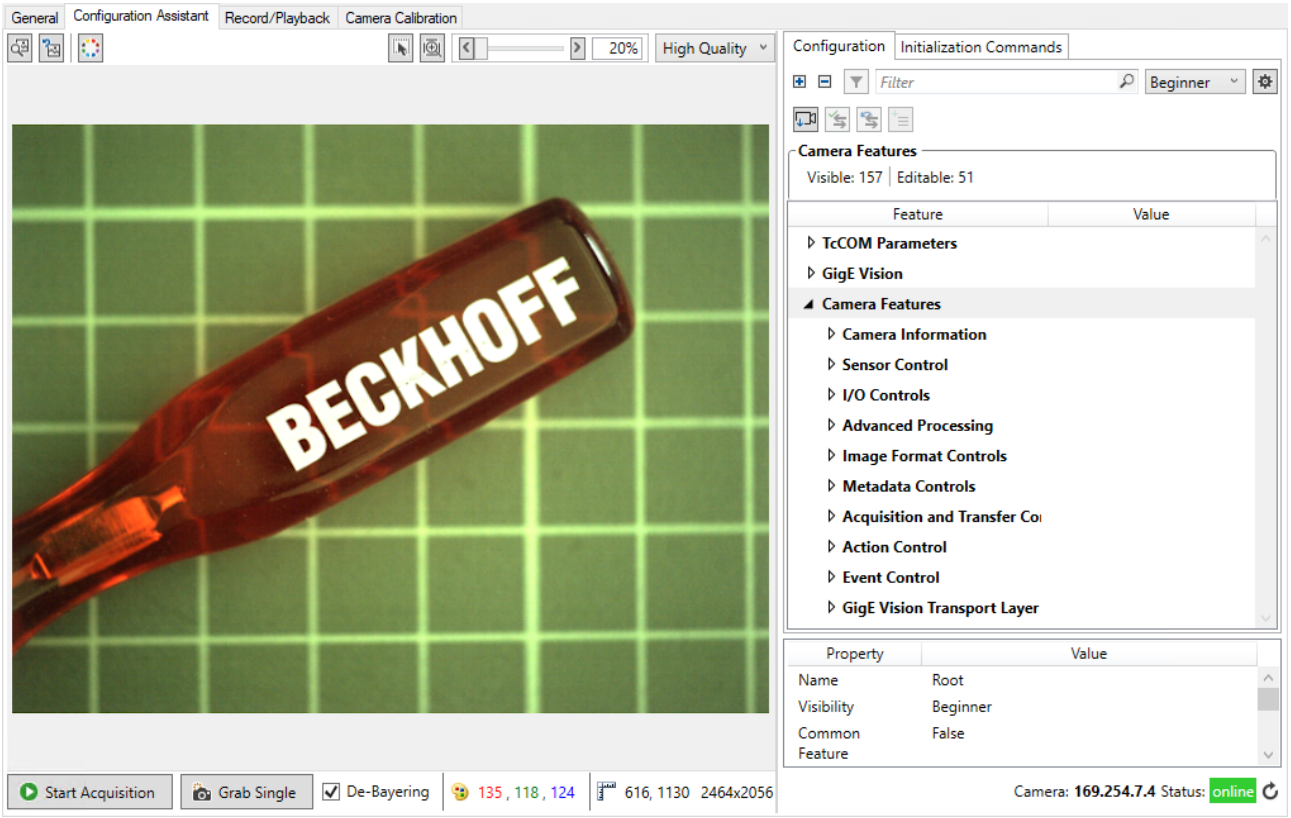
Im unteren Teil werden allgemeine Kamerainformationen angezeigt, soweit diese aus der Kamera ausgelesen werden konnten (u.a. Hersteller und Modellbezeichnung der Kamera). Diese Informationen sind ebenfalls im Konfigurationsbaum [▶ 83] zu finden.



5.4.2 Configuration Assistant

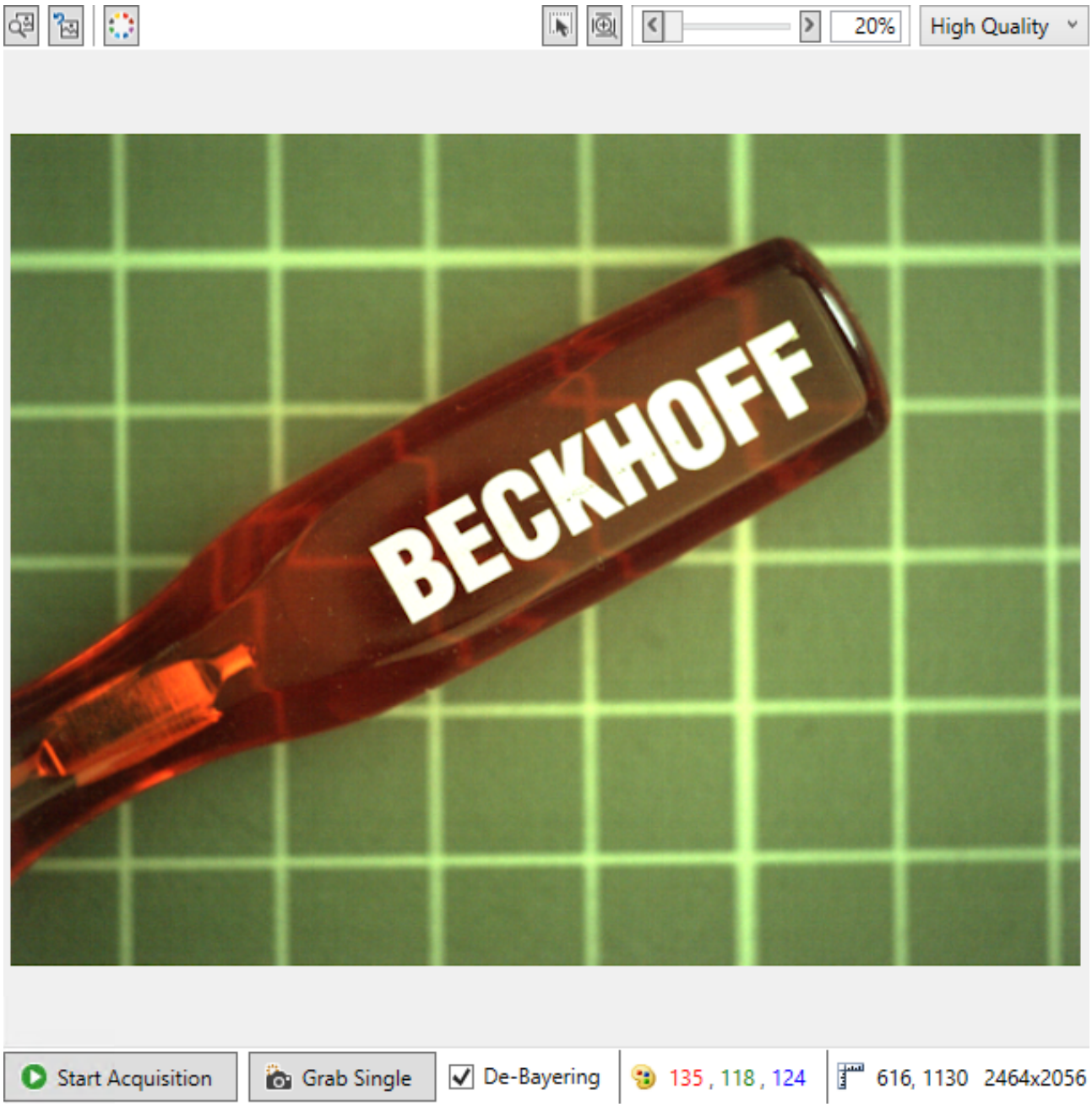
In dem Karteireiter **Configuration Assistant** können sowohl die Kameraparameter als auch das Kommunikationverhalten konfiguriert werden. Der Configuration Assistant lässt sich thematisch in vier Bereiche aufteilen:

- Bildvorschau [▶ 80]
- Konfigurationsbaum [▶ 83]
- Initialization Commands [▶ 94]
- Aktionen & Einstellungen [▶ 91]
- TcCOM-Parameter [▶ 106]



5.4.2.1 Bildvorschau

Im linken Bereich des Configuration Assistant befinden sich ein Vorschaufenster sowie einige Bedienelemente.



In dieser Bildvorschau werden nur 8-Bit Integer Bilder korrekt dargestellt.

Untere Leiste

Unterhalb des Vorschaufensters befinden sich die Controls zum Starten / Stoppen der Vorschau, sowie Informationen über die Cursorposition auf dem Bild:

Start/Stop Acquisition	Button zum Starten bzw. Stoppen der Bildaufnahme.
Grab Single	Button zum Triggern eines einzelnen Bildes.
De-Bayering	Bilder im Bayer-Format können alternativ als RGB-Bild in der Vorschau angezeigt werden. **Wenn diese Option aktiviert wird, wird erst das nächste Bild von der Kamera konvertiert, nicht das aktuelle.**
Pixel Value	Hier wird der Farb- bzw. Intensitätswert des Pixels an der Maus-Cursor-Position auf dem Bild wiedergegeben.
Cursor Position	Hier wird die Maus-Cursor Position auf dem Bild wiedergegeben. Dabei ist zu beachten, dass der Null-Punkt bei Bildern immer oben links ist.
Image size	Hier wird die Größe des angezeigten Bilds in Pixeln wiedergegeben.

Bildaufnahme- und Triggereinstellungen

Achten Sie darauf, dass die Bildaufnahme- und Triggereinstellungen der Kamera so gewählt sind, dass die Kamera auf die Kommandos reagiert.
Wenn z. B. der Trigger Mode der Kamera aktiv ist, wird die Kamera beim Starten der Bildaufnahme nicht automatisch Bilder senden. Siehe das Beispiel Bild-Akquise und Trigger [▶ 2936] für weitere Details.

Obere Leiste

Oberhalb des Vorschaubildes gibt es einige Controls zur Erleichterung der Kamerakonfiguration und Anpassung der Bildanzeige.

Kamerakonfiguration

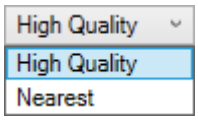
Set ROI	Nach Klick auf diesen Button kann in dem Vorschaubild direkt eine ROI durch Gedrückthalten der linken Maustaste gezogen werden. Diese ROI wird in den Kamera-Parametern geschrieben, sodass das nächste Bild mit der gesetzten ROI gesendet wird.
Reset ROI	Nach Klick auf diesen Button wird eine zuvor gesetzte ROI gelöscht, d. h. Bildgröße und Offset werden auf die Standardwerte der Kamera gesetzt.
Whitebalance	Durch Klick wird ein Weißabgleich bei Farbkameras ausgeführt. Die Kamera muss dafür die folgenden GenAPI-Parameter unterstützen: *BalanceRatioSelector* und *BalanceRatio* (bzw. *BalanceRatioAbs* bei älteren Kameras). Zudem muss das Kamerabild für einen Weißabgleich ein weißes (nicht überbelichtetes oder reflektierendes) Objekt enthalten; z. B. ein Blatt Papier. Nach einem Weißabgleich stehen die nötigen Werte in den genannten GenAPI-Parametern gespeichert und können in die Initialization Commands [▶ 94] oder UserSets übernommen werden. Eine automatisch persistente Speicherung auf der Kamera kann kamera-spezifisch vorkommen, ist jedoch nicht standardmäßig vorhanden. Dieser Button ist nur aktiv, wenn ein valides Farbformat vorliegt (3-kanalige Bilder als RGB oder 1-kanalige Bilder im Bayer-Format).

Angezeigte Bildgröße

Einstellmöglichkeit für die Anzeigegröße des Bildes. Das Bild lässt sich zwischen 20% und 6400% der Originalgröße über den Slider oder durch Eingabe eines Wertes im Textfeld skalieren. Nach Klick auf den linken Button kann in dem Vorschaubild direkt ein rechteckiger Bereich durch Gedrückthalten der linken Maustaste gezogen werden. Im Anschluss wird die Skalierung entsprechend angepasst, so dass der ausgewählte Bereich in der Vorschau angezeigt wird. Durch einen Klick auf den **„Fit to window“** Button, zweiter von links, wird die Anzeigegröße automatisch so gewählt, dass die Vorschau den verfügbaren Platz auf dem Bildschirm bestmöglich ausnutzt.

Interpolation

Hier kann zwischen verschiedenen Interpolationsmethoden gewählt werden, um das Bild aus der Originalgröße in die Anzeigegröße zu skalieren:



High Quality	Interpolation mit hoher Qualität, mit weichen Kanten- und Pixelübergängen.
Nearest	Interpolation durch Übernehmen des örtlich nächsten Pixelwertes, wodurch die einzelnen Pixel sichtbar und abgegrenzt dargestellt werden. Diese Methode bietet sich an um Pixelwerte zu überprüfen.

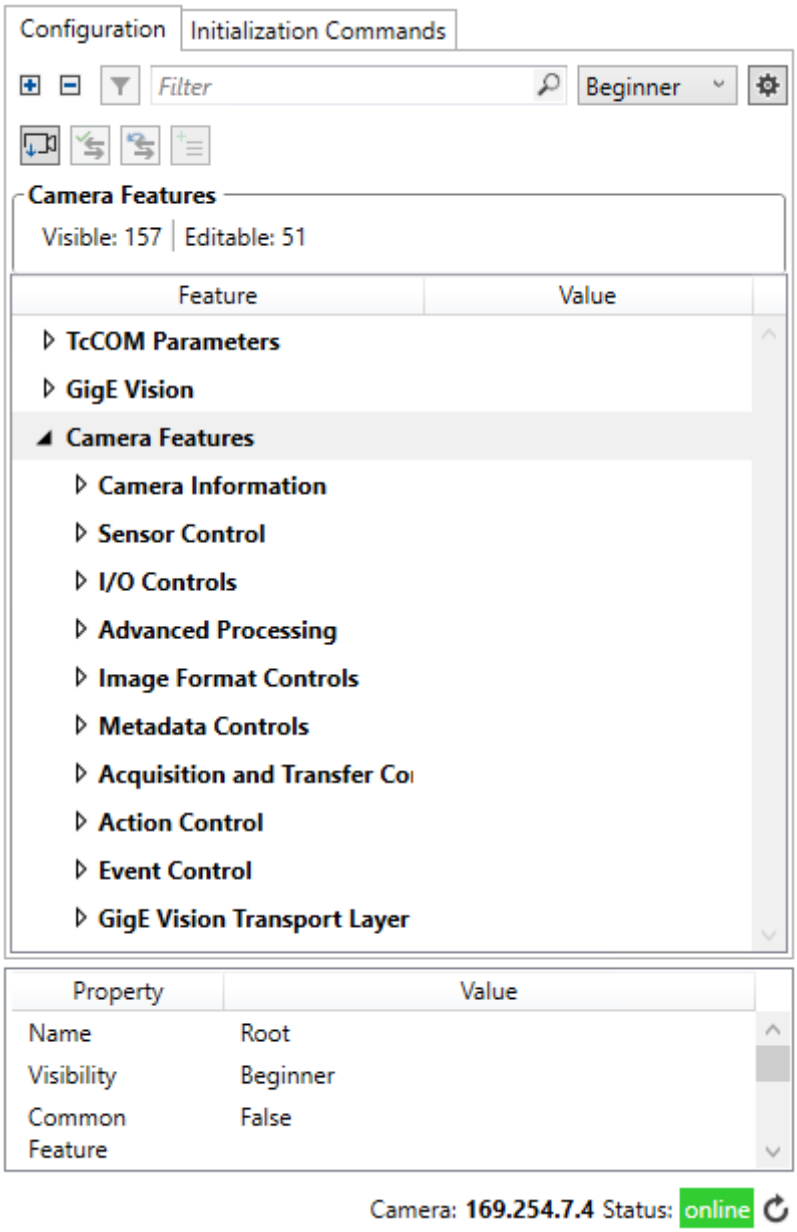
Rechtsklick auf das Bild

Start Acquisition	Bildaufnahme starten.
Stop Acquisition	Bildaufnahme stoppen.
Save Current Image	Bild als Bitmap, JPEG oder PNG abspeichern.

5.4.2.2 Konfiguration

Das Configuration Tab setzt sich aus 3 Bereichen zusammen, die im Folgenden einzeln beschrieben werden:

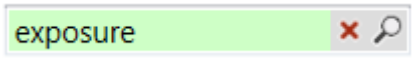
- Filter- und Bedienelemente
- Konfigurationsbaum
- Eigenschaftsfenster



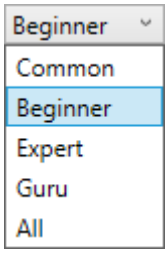
Filterelemente

Mit Hilfe von Filterelementen können Features im Baum der Konfiguration und in der Liste Initialization Commands schnell und einfach gefunden werden:

- Durch einen Klick auf das -Symbol werden sämtliche Baumelemente vollständig aufgeklappt. Alle Parameter sind sichtbar.
- Durch einen Klick auf das -Symbol werden sämtliche Baumelemente vollständig eingeklappt. Nur noch die Hauptgruppen sind sichtbar.
- Durch einen Klick auf das -Symbol werden die Baumelemente gefiltert und nur die Zeilen mit einer Änderung werden angezeigt. Der Button ist nur aktiv, wenn Änderungen angezeigt werden.

- Mit der Suchzeile 

werden die Parameter nach bestimmten Zeichenfolgen oder (hexadezimalen) Adressen gefiltert. Bei der Eingabe einer Zeichenfolge ist der Hintergrund des Textfeldes weiß und es erscheint ein zusätzlicher Button zum Löschen bzw. Zurücksetzen des Filters. Wenn eine Eingabe durch Enter oder Klick des Filter-Buttons bestätigt wurde, wird das Textfeld zusätzlich grün hinterlegt. Dadurch wird gekennzeichnet, ob die aktuelle Zeichenfolge dem aktiven Filter und somit der Ansicht der Baumelemente entspricht.
- Entsprechend des GenICam-Standards können die drei Sichtbarkeitslevel **Beginner**, **Expert** und **Guru** gewählt werden. Das Sichtbarkeitslevel jedes Features ist im GenApi-File der Kamera hinterlegt. Zusätzlich gibt es den Eintrag **All** bei dem alle Einträge inkl. der invisibel Features angezeigt. Zudem gibt es noch den Eintrag **Common,** mit dem nur die vor- bzw. selbstdefinierten-Features angezeigt werden.



Common Features Liste

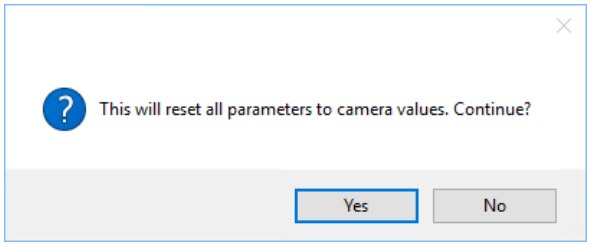
Die Common Features Liste ist eine vordefinierte Liste, die die gebräuchlichsten bzw. am häufigsten zur Konfiguration einer Kamera verwendeten Features enthält. Für die vordefinierten Register wird der Name aus der GenICam Standard Features Naming Convention (SFNC) verwendet. Daher kann sich die Anzahl unterscheiden, je nachdem welche Features die jeweilige Kamera grundsätzlich unterstützt und ob der Standardname verwendet wurde.

Die Liste der enthaltenen Features kann individuell erweitert oder reduziert werden. Um ein Kamera Feature hinzuzufügen, wählen Sie einen Eintrag aus und klicken den Button zum Hinzufügen. Wenn das Feature schon in der Liste enthalten ist, kann es über den Button entfernt werden. Alle Änderungen werden im TwinCAT Projekt gespeichert. Es können keine Kategorien und Selektoren hinzugefügt werden.

Über den Eintrag „Reset Common Features configuration“ im Aktionsmenu kann die Liste auf den vordefinierten Zustand zurückgesetzt werden. Die Liste mit der Bezeichnung „Common“ kann als Filter für die Konfiguration sowie auch für die Initialization Commands ausgewählt werden.

Bedienelemente

Die Funktionen der Buttons von links nach rechts sind:

Read Camera Parameters	Alle Parameterwerte werden aus der Kamera ausgelesen und in den Konfigurationsbaum geschrieben. Dies muss zunächst durch folgenden Dialog bestätigt werden: 
Write Local Changes	Alle lokalen Parameterwerte, die von den Kamerawerten abweichen (orange im Konfigurationsbaum gekennzeichnet), werden auf die Kamera geschrieben.
Revert Local Changes	Alle lokalen Änderungen im Feature-Baum, welche noch nicht auf die Kamera geschrieben wurden, werden zurückgesetzt.

Konfigurationsbaum

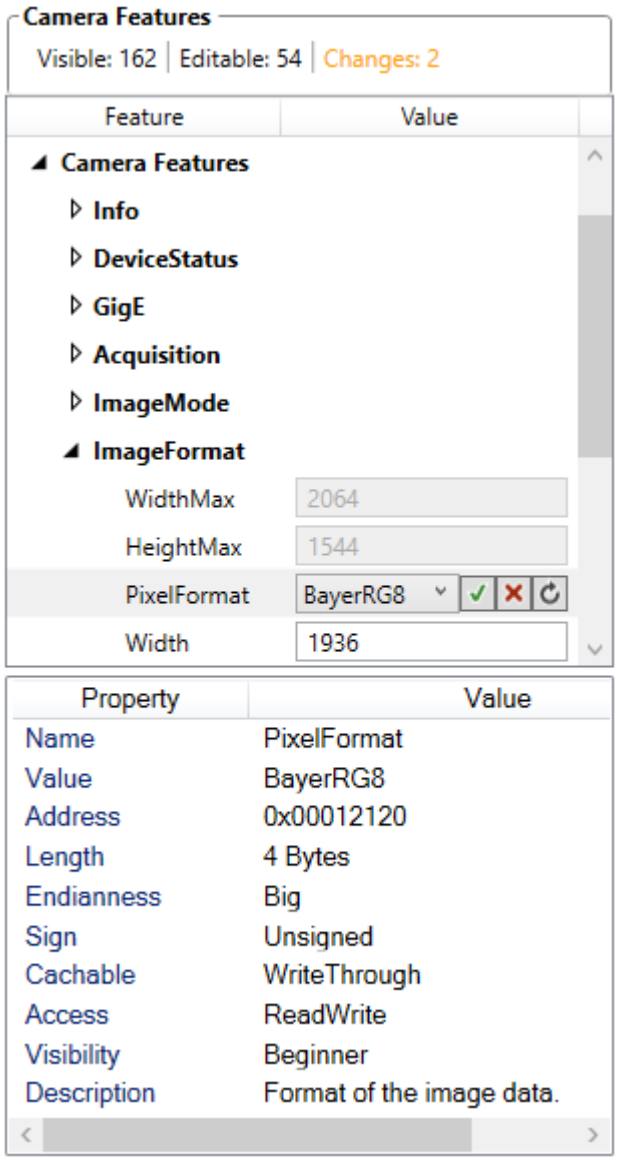
Der Konfigurationsbaum enthält drei Hauptgruppen:

- **TcCOM Parameters**: Parametrierung des Kommunikationsverhaltens zwischen TwinCAT und der angeschlossenen Kamera. Eine Erklärung dieser Parameter ist im Kapitel TcCOM-Parameter [▶ 106] zu finden.
- **GigE Vision**: Informationen zur Kamera entsprechend des GigE Vision Standards, sowie Möglichkeit der IP-Konfiguration. Die Einstellungen können – je nach Kamera – auch im Camera Features-Baum vorhanden sein.
- **Camera Features**: Parametrierung der Kamera entsprechend des GenICam Standards. Die Beschreibung aller Kamera-Features wird automatisiert aus der GenAPI-Datei von der Kamera ausgelesen. Ein Feature kann ein ausführbares Kommando, ein (änderbarer) Parameter oder ein Organisationselement sein. Für einige Einstellungen (z. B. Bild-Akquise/Trigger, Binning, Belichtungszeit, etc.) siehe Beispiele zur Kamera-Konfiguration [▶ 2932].
  Details zu den Parametern einer jeweiligen Kamera entnehmen Sie der entsprechenden Herstellerdokumentation. Zudem beschreibt der GenICam-Standard [▶ 132], wie der Camera Features-Baum strukturiert wird.

Camera Features Informationen

Oberhalb der Konfigurationstabelle werden drei Informationen zu den Camera Features angezeigt:

- Visible: Anzahl aller aktuell sichtbaren bzw. angezeigten Features, abhängig von den aktiven Filtereinstellungen.
- Editable: Anzahl der „Visible“-Features, die editierbar sind. Die Anzahl ist generell abhängig von der aktuellen Kamera Parametrierung und variiert daher.
- Changes: Gesamte Anzahl von Änderungen innerhalb des Camera Features-Baums. Diese Anzahl wird nur eingeblendet, wenn Änderungen vorhanden sind, unabhängig von den aktiven Filtereinstellungen.



Wenn Sie ein Feature im Konfigurationsbaum anklicken, werden unterhalb des Baums die Eigenschaften des Features angezeigt.

Einstellung von Parametern

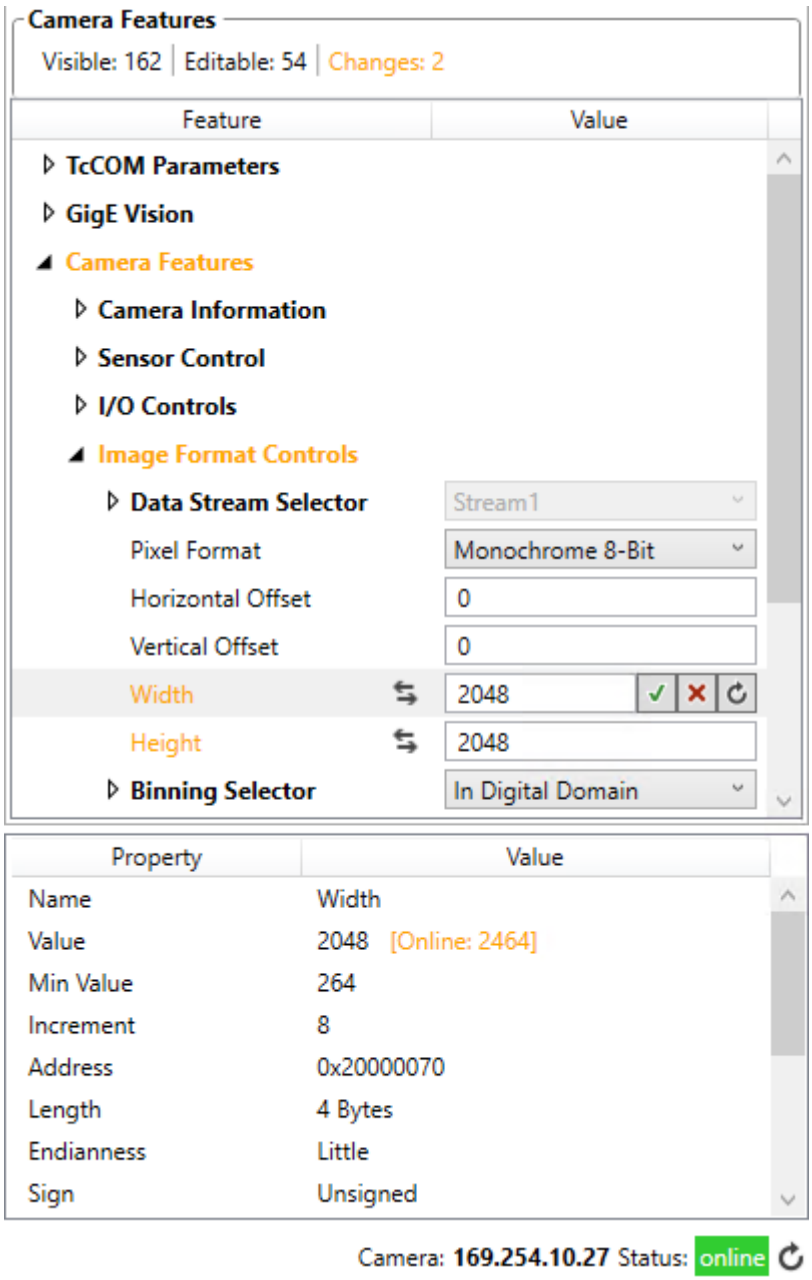
Änderbare Features können über ein Textfeld, einen Slider, eine Combobox oder eine Checkbox geändert werden. ReadOnly-Features sind ausgegraut. Für jedes änderbare Konfigurationselement stehen folgende Buttons zur Verfügung:

Write Value
Schreibt die lokale Änderung auf die Kamera.

Reset Value
Löscht die lokale Änderung, falls ein Unterschied zwischen lokalem und Kamerawert besteht.

Load Value
Lädt den aktuellen Wert des Parameters aus der Kamera.

Wenn der eingestellte Parameterwert vom tatsächlichen Wert auf der Kamera abweicht, werden der Parameter sowie Über-Elemente im Konfigurationsbaum in Orange markiert. Zusätzlich werden die Zeilen mit Änderungen mit einem Icon markiert und die Abweichung wird im Eigenschaftenfenster angezeigt. Ein oranger „Changes“ Hinweis oberhalb der Tabelle weist generell auf Änderungen in der Liste hin.



Offline-Änderungen

Die Parameter im Konfigurationsbaum sind auch änderbar, wenn die Kamera offline ist oder sich das Target im Run-Modus befindet. In dem Fall werden die Parameter nicht direkt auf die Kamera geschrieben, sondern nur im Projekt gespeichert. Dadurch sind offline Parameter-Änderungen möglich, die zu einem späteren Zeitpunkt mit der Kamera synchronisiert werden können.

Feature-Eigenschaften

Die Eigenschaften der einzelnen Features werden unterhalb des Konfigurationsbaums angezeigt. Dazu zählen u.a. die Größe und Adresse des jeweiligen Kamera-Registers, die Zugriffsrechte sowie eine Beschreibung.

Property	Value
Name	Width
Value	2464
Min Value	272
Increment	8
Address	0x20000070
Length	4 Bytes
Endianness	Little
Sign	Unsigned
Cachable	WriteThrough
Access	ReadWrite
Register Value	0xA0090000
Register Type	Integer
Visibility	Beginner
DisplayName	Width
Category	Image Format Controls
Common Feature	True
Description	Width of the Image provided by the device (in pixels).

Eigenschaft	Beschreibung
Name	Name des Parameters.
Beckhoff Tooltip	Enthält zusätzliche Informationen und wird nur in speziellen Fällen angezeigt. Z.B. sind die Acquisition Kommandos deaktiviert, da nur die Controls zum Starten / Stoppen der Vorschau verwendet werden sollen.
Value	Lokaler Wert des Parameters. Wenn der lokale Wert vom Kamerawert abweicht, wird zusätzlich der Kamerawert in Orange angezeigt.
Min Value	Minimaler Einstellwert. Wird nur angezeigt, wenn bei dem Feature ein min. Wert in der GenApi angegeben wurde.
Max Value	Maximaler Einstellwert. Wird nur angezeigt, wenn bei dem Feature ein max. Wert in der GenApi angegeben wurde.
Increment	Inkrement der Wertänderung. Wird nur angezeigt, wenn bei dem Feature ein min. Wert in der GenApi angegeben wurde.
Address	Hexadezimale Adresse des Parameters auf der Kamera
Length	Größe des Parameters in Bytes
Endianness	Byte-Reihenfolge des Parameters. Diese Eigenschaft muss bei der Benutzung der Funktionsblöcke FB_VN_ReadRegister [▶ 1624] und FB_VN_WriteRegister [▶ 1632] beachtet werden.
Sign	• **Signed**: Parameter ist Vorzeichen-behaftet • **Unsigned**: Parameter ist nicht Vorzeichen-behaftet
Cachable	Beschreibt, ob der Parameter-Wert gespeichert („gecached“) wird, wenn keine Kamera verbunden ist. • **WriteThrough**: Der Wert wird sowohl in die Kamera als auch in den Projekt-Cache und Projekt-Speicher geschrieben. • **NoCache**: Der Wert dieses Kamera-Parameters soll laut GenAPI-Beschreibung der Kamera nicht lokal gespeichert werden. Daher kann dieser Parameter bei fehlender Kamera-Verbindung leer sein.
Access	Zugriffsrechte zum Lesen und Schreiben dieses Parameters auf der Kamera.
RegisterValue	Register-Wert des Parameters. Dieser kann vom angezeigten Wert abweichen, wenn es sich um bekannte Typen handelt. Beispiel: Name: PixelFormat Value: BayerRG8 RegisterValue: 13
Bitmask	Verwendete Bitmaske, wird nur bei maskierten Registern angezeigt.
RegisterType	Datentyp des Registers. An diesem Typ kann abgelesen werden, welcher der Read- / Write-Funktionsblöcke zu verwenden ist.
Visibility	Sichtbarkeitsstufe des Parameters. Mögliche Werte: **Beginner**, **Expert, Guru** und **All** Die Sichtbarkeit von Parametern kann mit den Filterelementen [▶ 84] eingestellt werden.
DisplayName	Anzuzeigender Name des Features.
Category	Name der Kategorie in die das Feature einsortiert ist.
Common Feature	Gibt an, ob das Feature in der Common Feature List enthalten ist.
Description	Bedeutung des Parameters

Diese Eigenschaften werden z. B. benötigt, wenn aus der SPS heraus ein Parameter ausgelesen oder verändert werden soll. Im Wesentlichen muss dabei auf Address, Length, Endianness und RegisterValue geachtet werden. Die entsprechenden Funktionsblöcke finden Sie in der API-Referenz unter Camera Register Access [▶ 1621].

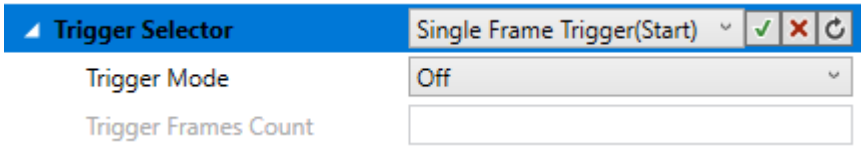
Abhängig vom Datentyp eines Registers können noch weitere Eigenschaften wie Min/Max/Inkrement hinzukommen.

Selektoren

Selektoren sind Parameter, die zur Gruppierung und Auswahl von mehreren Unter-Features dienen. Um ein Unter-Feature zu bedienen, muss zunächst der Selektor-Parameter passend eingestellt werden (sowohl im Konfigurationsassistenten als auch in der SPS).

Wenn ein Selektor-Parameter keine Adress-Eigenschaft besitzt, zeigt dieser nicht auf ein Kamera-Register. Er dient dann lediglich der übersichtlicheren Anzeige im Konfigurationsassistenten. In diesem Fall können die Unter-Parameter aus der SPS direkt geändert werden.

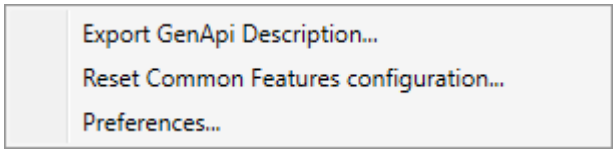
Um beim Erstellen der Initialization Commands alle Unter-Parameter eines Selektors zu beachten, wird standardmäßig automatisch über alle Unter-Features iteriert. Eine entsprechende Einstellung finden Sie unter Aktionen & Einstellungen [▶ 91]. Ein Beispiel für einen Selektor ist der Trigger-Selector:



5.4.2.3 Aktionen & Einstellungen

Aktionen

Oben rechts befindet sich der Action-Button , über den weitere Aktionen ausgeführt werden können bzw. zusätzliche Einstellungen möglich sind:

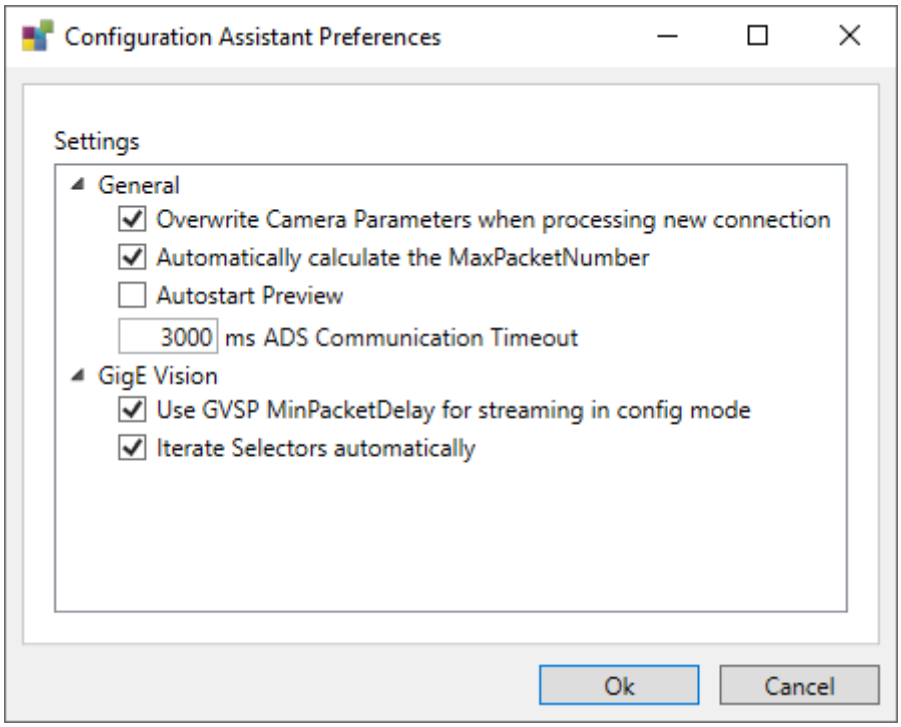


Export GenApi Description...	Die GenAPI-Beschreibung der Kamera wird lokal in einem anzugebenden Pfad gespeichert.
Reset Common Features configuration…	Setzt die Common Features Liste auf den vordefinierten Stand zurück.
Preferences...	Das folgend beschriebene Einstellungsfenster wird geöffnet.

Einladen gespeicherter GenAPI-Datei nicht möglich

Das Einladen einer gespeicherten GenAPI-Datei ist nicht möglich, da zum Auslesen einiger Kamera-Features eine Verbindung zur Kamera benötigt wird.

Einstellungen



Overwrite Camera Parameters when loading Project	Mit dieser Einstellung wird gesteuert, ob gespeicherte Kamerawerte beim Laden des Projekts direkt auf die Kamera geschrieben werden sollen. Dadurch ist die Kamera auch beim Konfigurieren automatisch in dem Zustand, mit dem das Projekt zuletzt gespeichert wurde. Dies funktioniert nur beim ersten Initialisieren der Kamera-Objekte, nachdem das Projekt geladen wurde, sofern die Kamera zu dem Zeitpunkt verbunden ist und sich TwinCAT dabei im Config-Modus befindet. Standard: aktiviert
Automatically calculate the MaxPacketNumber	Wenn diese Option aktiv ist, wird nach dem Auslesen aller Kameraparameter automatisch die Anzahl an maximal benötigten (und erlaubten) Paketen für ein Bild ermittelt. Die Berechnung geschieht auf Basis von Bildgröße und Pixelformat. Der errechnete Wert wird nur dann in das zuständige GVSP-Objekt geschrieben, sofern er größer als der aktuell eingestellte ist. Diese Funktionalität dient der automatischen Anpassung des reservierten Speicherplatzes für ein zu empfangendes Bild an die tatsächliche Menge an Bilddaten. Standard: aktiviert.
Autostart Preview	Wenn diese Option aktiv ist, wird automatisch beim Öffnen des Configuration Assistent [▶ 79] das Livebild der Kamera angezeigt, soweit dies möglich ist. Standard: deaktiviert.
ADS Communication Timeout	Der Communication Timeout betrifft die Kommunikation innerhalb der Kamerainstanz in TwinCAT. Er muss nur dann erhöht werden, wenn auch andere Timeouts im GVCP oder GVSP Modul erhöht werden müssen. Einheit: Millisekunden (ms) Standard: 3.000 (angepasst an Kamera)
Use GVSP MinPacketDelay for streaming in config mode	Wenn diese Option aktiv ist, wird zur Reduzierung der Bandbreite im Configuration Mode das MinPacketDelay aus dem GVSP Module verwendet. Wenn Sie diese Option ausschalten, wird stattdessen die Frame Rate der Kamera reduziert. Standard: aktiviert.

Iterate Selectors automatically	Wenn diese Option aktiv ist, werden Kamera-Enumerationsparameter, die als Selektoren fungieren, automatisch iteriert. Der Wert eines Selektor-Parameters wird dabei als Index zur Adressierung anderer Parameter benutzt. Durch eine Änderung eines Selektor-Werts können die abhängigen Parameter entsprechend geändert werden, wodurch eine konsistente Parametrierung der Kamera erreicht wird. Ohne diese Iteration würden nur die Parameter des aktuell ausgewählten Selektors beschrieben werden. Dies gilt sowohl für die Einstellungen im Konfigurationsbaum [▶ 83] als auch beim Erstellen der Initialization Commands [▶ 94]. Standard: aktiviert.

HINWEIS

Einfluss auf Initialization Commands

Wenn die Option **Iterate Selectors automatically** verändert wird, müssen anschließend die Initialization Commands [▶ 94] neu erstellt werden. Erstellen Sie die Initialization Commands neu, indem Sie auf dem Tab Initialization Commands den Button zum Erstellen anklicken.

5.4.2.4 Initialization Commands

Die Initialization Commands erfüllen den Zweck, eine bestimmte Parameter-Konfiguration der Kamera abzuspeichern und wenn erforderlich, die Kamera auf ebendiese Konfiguration einzustellen. Diese Funktionalität wird benötigt, da Kameras nach einem Neustart im Standard den Auslieferungszustand widerspiegeln. Die Initialisierung kann entweder automatisch beim Starten von TwinCAT oder manuell durch einen Methoden-Aufruf in der SPS geschehen.

Die Initialization Commands sind vergleichbar mit der Startup-Liste von EtherCAT-Devices oder mit UserSets. Das UserSet ist ein GenAPI Feature, dass das Abspeichern und automatische Laden einer Konfiguration in der Kamera ermöglicht. UserSets sind nach dem GigE Vision Standard nicht verpflichtet und werden daher nicht in allen Kameras nativ bereitgestellt.

Um solch eine Funktionalität dennoch für alle GigE Vision Kameras einheitlich bereitstellen zu können, bietet TwinCAT Vision die Möglichkeit mit den Initialization Commands. Wenn vorhanden, können UsetSets selbstverständlich auch als alternative genutzt werden.

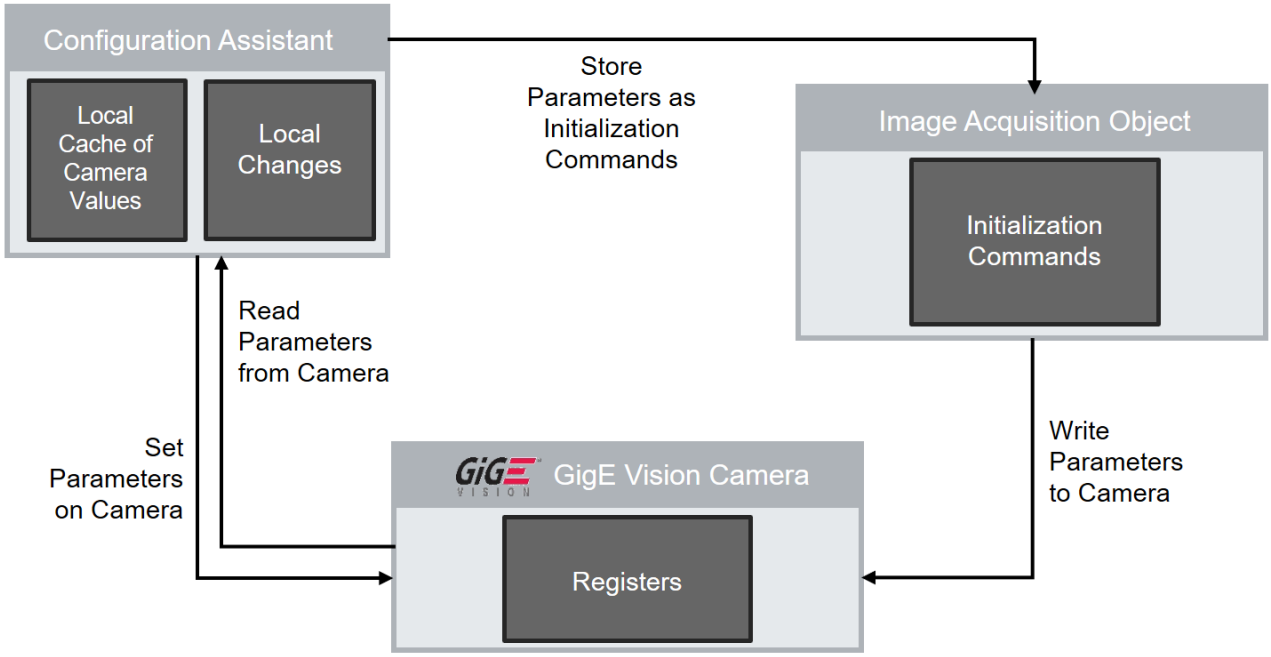
Die Initialization Commands lassen sich neben diesen allgemeinen Informationen thematisch in folgende Bereiche aufteilen:

- Konfiguration [▶ 95]
- Erstellung [▶ 100]
- Problembehandlung [▶ 102]

Kamera-Zustand nach einem Bootvorgang

Nach dem Booten einer GigE Vision Kamera, enthält diese in der Konfiguration im Standard die Fabrikeinstellungen. Um einen vorherigen Konfigurationszustand herzustellen, müssen die entsprechenden Parameter über UserSets, über Initialization Commands oder im Hintergrund durch den Kamera-Assistenten zunächst auf die Kamera geschrieben werden

Die Rolle der Initialization Commands lässt sich wie folgt beschreiben:



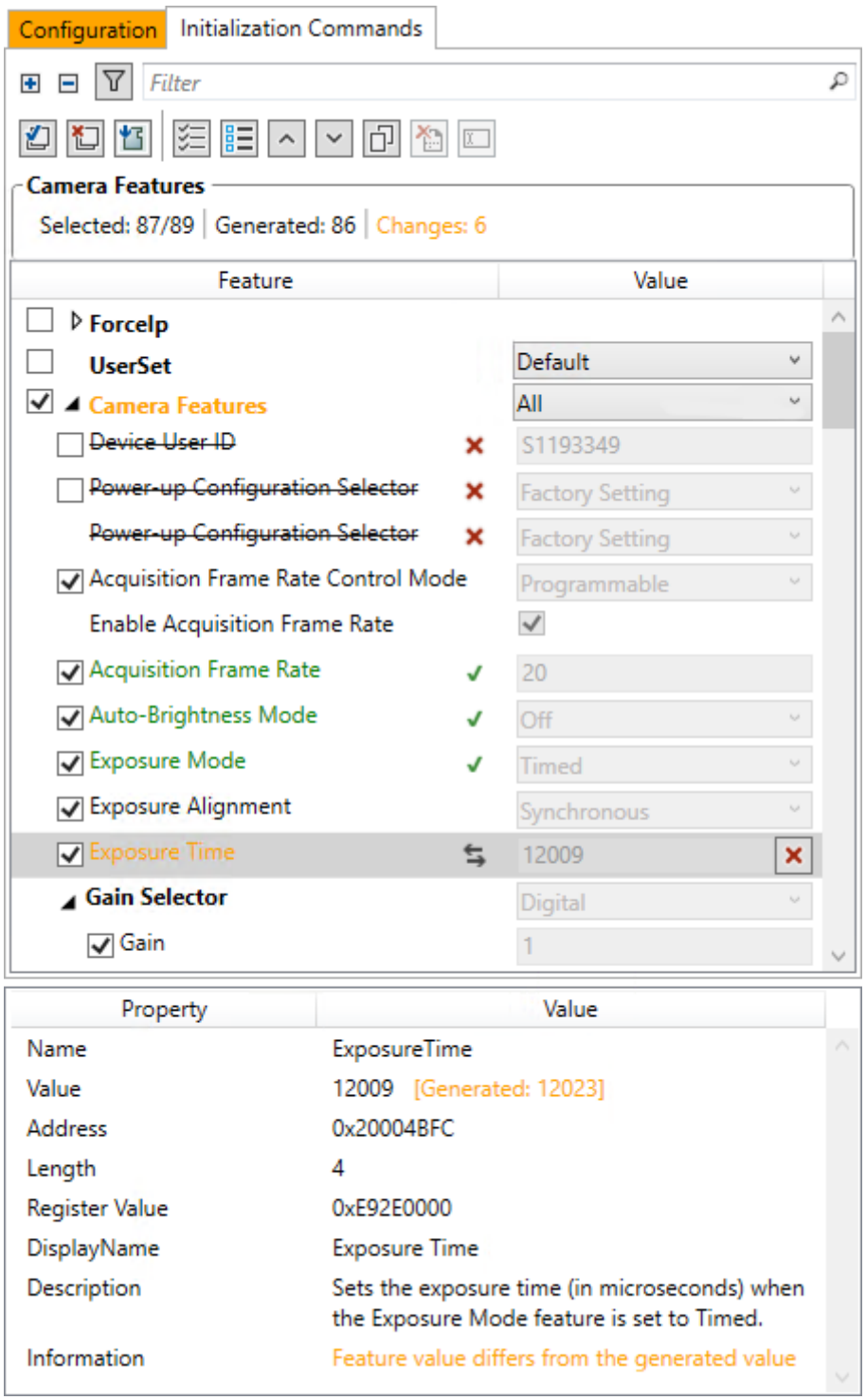
Dabei werden die Lese- und Schreibvorgänge wie folgt ausgelöst:

Aktion	Auslöser
Parameter von der Kamera lesen	• Initiales Auslesen nach Anlegen der Kamera • Auslesen beim Neustarten der TcCOM-Objekte (z. B. beim Klick auf **Load Devices**) • Klick auf **Read Camera Parameters**
Schreiben von einzelnen Parametern auf die Kamera	• Einzelnes Verändern und Schreiben der Parameter durch Klick auf **Write Value** im Konfigurationsbaum [▶ 83]. • Klick auf **Write Local Changes** • Bei der ersten Kamera-Verbindung nach Laden des TwinCAT-Projekts, sofern **Overwrite Camera Parameters when loading Project** in den Einstellungen [▶ 91] aktiviert ist
Initialization Commands im Kamera-Objekt aus aktuellen Parameterwerten erstellen	• Klick im Initialization Commands Tab auf **Create and Test the selected Initialization Commands**
Parameter aus den Initialization Commands auf die Kamera schreiben	• Wenn das Image Acquisition TcCOM-Objekt in den OP-Zustand geschaltet wird (Start von TwinCAT), abhängig vom Parameter `InitializationAutoMode` im Image Acquisition [▶ 107] Objekt. • Aufruf über eine Methode des Funktionsbausteins FB_VN_GevCameraControl [▶ 1672]

5.4.2.4.1 Konfiguration

Das Initialization Commands Tab setzt sich aus 3 Bereichen zusammen, die im Folgenden einzeln beschrieben werden:

- Filter- und Bedienelemente
- Konfigurationsliste
- Eigenschaftsfenster

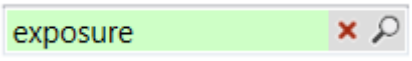


Filterelemente

Mit Hilfe von Filterelementen können Features im Baum der Konfiguration und in der Liste Initialization Commands schnell und einfach gefunden werden:

- Durch einen Klick auf das -Symbol werden sämtliche Baumelemente vollständig aufgeklappt. Alle Parameter sind sichtbar.
- Durch einen Klick auf das -Symbol werden sämtliche Baumelemente vollständig eingeklappt. Nur noch die Hauptgruppen sind sichtbar.

- Durch einen Klick auf das -Symbol werden die Baumelemente gefiltert und nur die Zeilen mit einer Änderung werden angezeigt. Der Button ist nur aktiv, wenn Änderungen angezeigt werden.

- Mit der Suchzeile 

werden die Parameter nach bestimmten Zeichenfolgen oder (hexadezimalen) Adressen gefiltert. Bei der Eingabe einer Zeichenfolge ist der Hintergrund des Textfeldes weiß und es erscheint ein zusätzlicher Button zum Löschen bzw. Zurücksetzen des Filters. Wenn eine Eingabe durch Enter oder Klick des Filter-Buttons bestätigt wurde, wird das Textfeld zusätzlich grün hinterlegt. Dadurch wird gekennzeichnet, ob die aktuelle Zeichenfolge dem aktiven Filter und somit der Ansicht der Baumelemente entspricht.

Bedienelemente

Die linke Gruppe der Buttons stellt die Grundfunktionalität bereit. Die Funktionen von links nach rechts sind:

Button	Beschreibung
Create and test the selected Initialization Commands	Mit den aktuellen Werten der Konfigurationsliste werden die Initialization Commands erstellt. **Dabei werden die Parameter ebenfalls an die Kamera geschrieben und zum Test zurückgelesen.**
Delete changes and Initialization Commands	Die erstellten und im TcCOM-Objekt hinterlegten Initialization Commands werden gelöscht. Das betrifft auch alle weiteren Änderungen, die in der Liste vorgenommen wurden. **Die gesamte Liste wird dadurch in den Initialzustand zurückversetzt.**
Apply generated Initialization Commands	Die erstellten und im TcCOM-Objekt hinterlegten Initialization Commands werden an die Kamera geschrieben.

Die rechte Gruppe der Buttons stellt spezielle Funktionalität bereit, die nur für die Einträge unter dem Camera Register Knoten aktiv sind. Die Funktionen von links nach rechts sind:

Button	Beschreibung
Select all features of the CameraRegister category	Alle Einträge unterhalb der Camera Register Kategorie werden angewählt.
Deselect all features of the CameraRegister category	Alle Einträge unterhalb der Camera Register Kategorie werden abgewählt.
Move element up	Das ausgewählte Element wird eine Listenposition nach oben verschoben.
Move element down	Das ausgewählte Element wird eine Listenposition nach unten verschoben.
Copy element	Von dem ausgewählten Element wird eine Kopie erzeugt. Der eingestellte Wert wird dabei ebenfalls kopiert. **Der Wert des kopierten Eintrags kann nachträglich nicht mehr geändert werden.**
Remove copied element	Entfernt ein zuvor kopiertes Element.
Rename copied element	Öffnet ein Fenster zum Umbenennen des kopierten Elements.

Konfigurationsliste

Die Konfigurationsliste enthält drei Hauptgruppen:

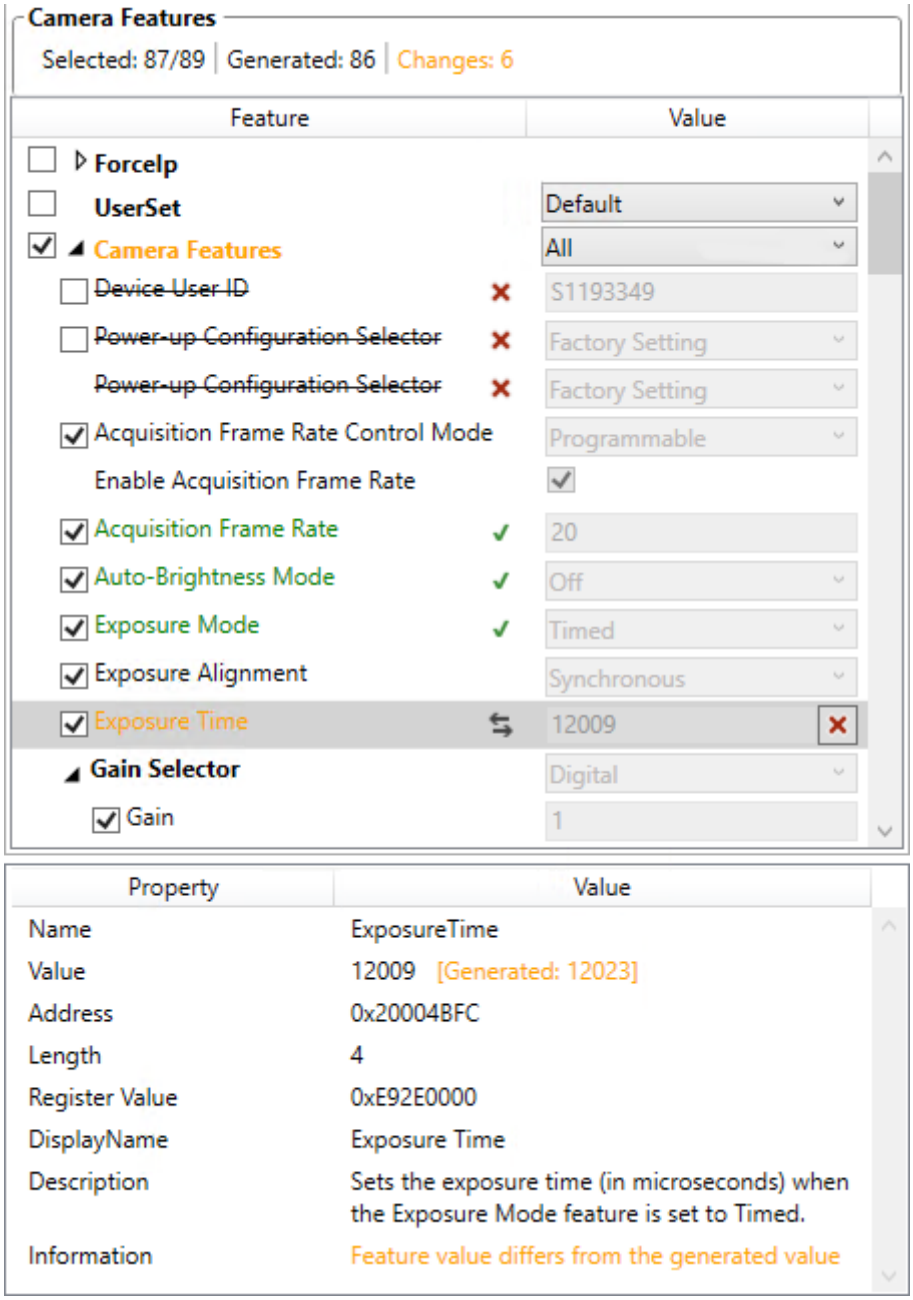
- **ForceIP**: Wenn diese Option aktiv ist, wird ein ForceIp bei der Initialisierung der Kamera in der Runtime durchgeführt. Dabei wird die CameraIPAddress aus dem Image Acquisition Module [▶ 107] verwendet. Da das Kommando an die jeweilige Mac Adresse der Kamera gesendet wird, müssen bei einem Kameratausch die Initialization Commands mit der geänderten Adresse neu generiert werden. Die verwendeten Werte werden hier nur angezeigt. Standard: deaktiviert.
- **UserSet**: Einige Kameras besitzen eine UserSet-Funktion, mit der ein in der Kamera gespeichertes Abbild von Parametern geladen werden kann, um damit die Kamera zu initialisieren. Sofern die Kamera diese Funktion bereitstellt, kann das gewünschte UserSet hier ausgewählt werden. Wenn die Option aktiv ist, werden die Kommandos zum Auswählen und Laden des entsprechenden UserSets in die Initialization Commands geschrieben. Standard: deaktiviert.
- **Camera Features:** In dieser Gruppe werden die Register abhängig von dem ausgewählten Filter angezeigt. Bei **All** werden alle Register aufgelistet, die in der ausgelesenen GenApi Datei als schreibbar deklariert und in der aktuellen Kamera-Konfiguration verfügbar sind. Das gilt unabhängig von der hinterlegten Sichtbarkeit, daher werden auch invisible Register aufgelistet, die im Konfigurationsbaum nur bei dem Sichtbarkeitslevel All angezeigt werden. Beim Eintrag **Common** werden alle schreibbaren Register aus der Common Feature List, inkl. der benötigten Selektoren, angezeigt. Die initiale Reihenfolge entspricht der Reihenfolge in der GenApi Datei. Wenn die Camera Features Option aktiv ist, werden alle darunter angewählten Register für die Erstellung der Initialization Commands verwendet. Die einzelnen Register können an- und abgewählt und über die Bedienelemente entsprechend verändert werden. Register, die die gleiche Adresse haben und sich nur durch eine Maskierung unterscheiden, werden nur mit einer Checkbox dargestellt, da sie nicht unabhängig voneinander geschrieben werden können. Änderungen gegenüber den erstellten Initialization Commands werden farblich und grafisch kenntlich gemacht. Standard: aktiviert. Über den **Reset Value**-Button können einzelne Wertänderungen auf die generierten Werte zurückgesetzt werden. Um alle Werte auf den generierten Stand zurückzusetzen kann der **Apply generated Initialization Commands**-Button verwendet werden.

Bei An- oder Abwahl einer der Gruppen oder Änderung einer Einstellung unterhalb der Gruppen müssen die Initialization Commands neu erstellt werden. Bei der Verwendung von UserSets enthalten die Initialization Commands anstatt der Parameter die notwendigen UserSet Kommandos. Die Optionen UserSet und CameraRegister schließen sich gegenseitig aus.

Camera Features Informationen

Oberhalb der Konfigurationsliste werden drei Informationen zu den Camera Features angezeigt:

- Selected: Anzahl aller aktuell ausgewählten Camera Features von der maximal verfügbaren Feature Anzahl. Diese ist generell abhängig von der aktuellen Kamera Parametrierung und variiert daher.
- Generated: Anzahl der zuletzt erstellten Initialization Commands, die im Projekt hinterlegt sind.
- Changes: Gesamte Anzahl von Änderungen innerhalb der Camera Features-Liste. Diese Anzahl wird nur eingeblendet, wenn Änderungen vorhanden sind, unabhängig von den aktiven Filtereinstellungen.

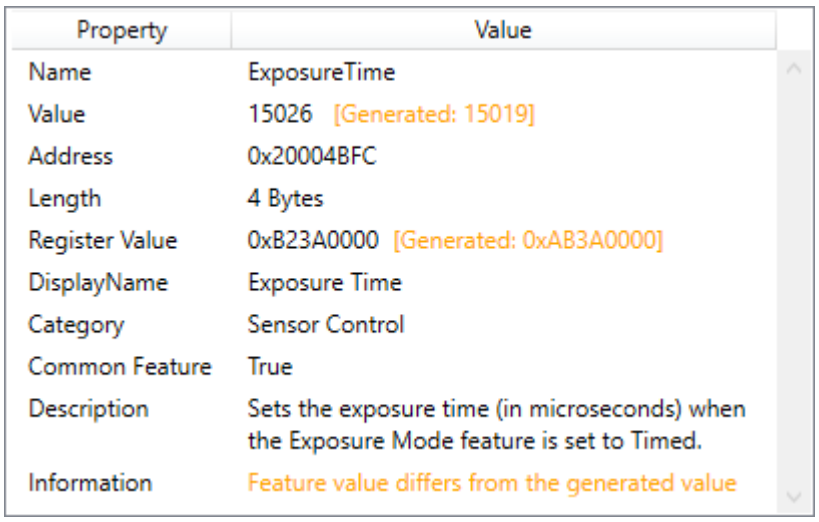


Selector Feature

Wenn **Iterate Selectors automatically** in den Einstellungen [▶ 91] aktiv ist, gibt es für jeden Wert eines Selector Features eine eigene Gruppe. Innerhalb dieser Gruppe werden alle von dem Selector Wert abhängigen und schreibbaren Register angezeigt. Daher können die Gruppen- und Registernamen mehrfach auftreten, wobei sich dann die Werte oder Registeradressen unterscheiden. Wenn die Einstellung nicht aktiv ist, gibt es pro Selector nur einen Eintrag.

Eigenschaftsfenster

Im Eigenschaftsfenster werden folgende Informationen zu dem ausgewählten Feature angezeigt:



Eigenschaft	Beschreibung
Name	Name des Parameters.
Value	Lokaler Wert des Parameters. Wenn der lokale Wert vom erstellten Initialization Commands-Wert abweicht, wird das Feature in Orange angezeigt.
Address	Hexadezimale Adresse des Parameters auf der Kamera.
Length	Größe des Parameters in Bytes.
RegisterValue	Register-Wert des Parameters.
Command Includes	Liste der Features, die in einem Schreibbefehl zusammengefasst werden. Z.B. maskierte Register, die daran zu erkennen sind, dass unter einer Checkbox zwei oder mehr Einträge zu finden sind.
DisplayName	Anzeigename des Features.
Category	Name der Kategorie in die das Feature einsortiert ist.
Common Feature	Gibt an, ob das Feature in der Common Feature List enthalten ist.
Description	Bedeutung des Parameters.
Information	Information, ob der Wert abweicht (orange), das Feature hinzugefügt (grün) oder entfernt (schwarz, durchgestrichen) wird.

5.4.2.4.2 Erstellung

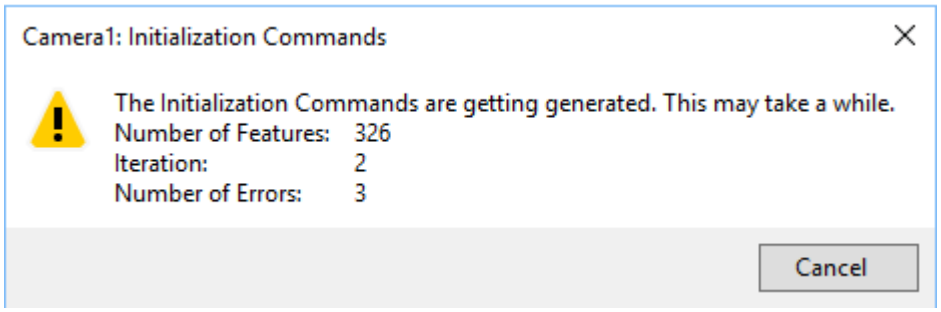
Das Erstellen der Initialization Commands kann nur über den Button **Create and test the selected Initialization Commands** auf dem Initialization Commands Tab ausgelöst werden. Zuvor sollten die Einstellungen bzw. Änderungen dahingehend überprüft werden, ob alle benötigten Register angewählt oder die nicht benötigten abgewählt sind und die Registerwerte den gewünschten Einstellungen für den Betrieb entsprechen.

Erstellungsschritte

Das Erstellen der Initialization Commands beinhaltet mehrere Aktionen, die im Hintergrund ausgeführt werden:

- Testweises Schreiben der Initialization Commands auf die verbundene Kamera.
- Testweises Lesen der zuvor geschriebenen Initialization Commands von der verbundenen Kamera.
- Falls Wertunterschiede oder Fehler festgestellt werden, erscheint eine Liste mit detaillierten Informationen. Falls keine kritischen Fehler aufgetreten sind, kann der Anwender die fehlerfreien Initialization Commands trotzdem übernehmen.
- Schreiben des Initialization Commands Arrays in das ImageAcquisition-TcCOM-Objekt der Kamera.

Durch das testweise Schreiben und anschließendes Auslesen wird validiert, ob die Kamera die Initialization Commands akzeptiert. Während der Generierung wird eines der beiden folgenden Fenster anzeigt:



Dabei werden folgende Statusinformationen angezeigt:

Number of Features	Anzahl der Parameter, mit denen die Initialization Commands erstellt wurden.
Iteration	Anzahl der bisherigen Versuche, die Initialization Commands auf die Kamera zu schreiben. Im Erfolgsfall ist dies „1“. Im Fehlerfall erhöht sich dieser Wert, da versucht wird, die Initialization Commands nochmal, ohne die fehlerhaften Parameter, erfolgreich auf die Kamera zu schreiben. Normalerweise sollten nicht mehr als 2 Iterationen auftreten; falls doch, muss die Concatination in den TcCOM-Parametern der ImageAcquisition deaktiviert werden.
Number of Errors	Anzahl der aufgetretenen Fehler. Diese werden im nachfolgenden Fenster genauer beschrieben.

Falls das Schreiben der Initialization Commands auf die Kamera sehr schnell erfolgreich funktioniert, verschwindet das Fenster sofort wieder und ist ggf. nicht sichtbar.

Meldungen

Nach dem erfolgreichen Erstellen der Initialization Commands besitzt die Kamera durch das testweise Schreiben exakt die Konfiguration, wie sie später während des Betriebs auf die Kamera geschrieben wird. Falls Fehler beim testweisen Schreiben der Initialization Commands auf die Kamera auftreten, werden diese im folgenden Fenster angezeigt:

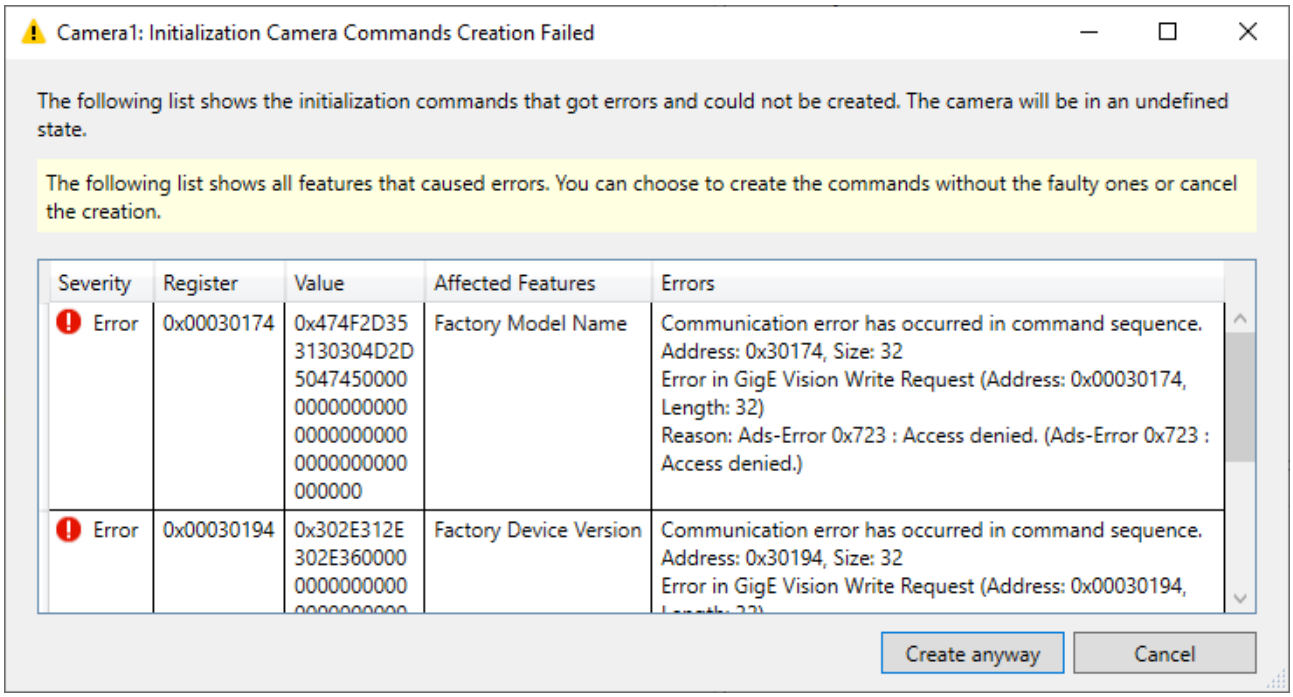


Abb. 1: Unkritische Fehler bei Erstellung von Initialization Commands

Sofern kein FATAL Fehler aufgetreten ist, können die Initialization Commands mit einem Klick auf **Create anyway** trotzdem erstellt und übernommen werden. In diesem Fall werden die fehlerhaften Parameter aus den Initialization Commands ausgeschlossen und dementsprechend später nicht auf die Kamera geschrieben.

Bei einem FATAL Fehler können die Initialization Commands nicht erstellt werden. Daher kann nicht auf **Create anyway** geklickt werden. Zunächst muss dann die Ursache des Fehlers behoben werden.

Unter Problembehandlung [▶ 102] sind diese Fehlerfälle sowie ihre Behebung beschrieben.

Warnungen beim Erstellen der Initialization Commands

Neben Fehlern können auch Warnungen auftreten. Diese sind meist nur ein Indiz dafür, dass die Kamera einen Wert nicht akzeptiert und korrigiert hat.

5.4.2.4.3 Problembehandlung

Bei der grundsätzlichen Kamerakommunikation und -parametrierung sowie der Erstellung der Initialization Commands sind hauptsächlich drei Komponenten relevant:

- der Inhalt der GenApi Gerätebeschreibungsdatei
- das Kameraverhalten
- und das GevImageAcquisition [▶ 133]-Objekt Verhalten.

GenApi Gerätebeschreibungsdatei

Die GigE Vision [▶ 131] Spezifikation beinhaltet das GenApi Module der GenICam-Spezifikation, in der die von einer Kamera unterstützten Funktionen beschrieben sind. Diese Beschreibung erfolgt in Form einer XML-Gerätebeschreibungsdatei, die konform zur Spezifikation erstellt sein muss. Die von einer Kamera bereitstellte GenApi-Beschreibung wird nach dem Empfang geprüft. Falls nicht konforme Stellen enthalten sind, werden diese als Warnung oder Fehler gemeldet.

- Eine Warnung erfolgt z. B. wenn sich nicht definierte Register im reservierten Bootstrap-Bereich befinden. Diese Register werden nachfolgend ignoriert, was in der Regel aber nicht zu Einschränkungen im Betrieb führt. Im Einzelfall ist zu prüfen, ob die gemeldeten Register benötigt werden.
- Ein Fehler wird z. B. gemeldet, wenn die beschriebenen Zugriffsrechte nicht mit dem Kameraverhalten übereinstimmen. Also wenn die Kamera beim Auslesen von lesbar markierten oder beim Schreiben von schreibbar markierten Registern einen Fehler zurückgibt.
- Falls die GenApi Beschreibung komplett nicht konforme Stellen beinhaltet, kann es in seltenen Fällen dazu kommen, dass die Kamera gar nicht betrieben werden kann.

Kameraverhalten

Das Kameraverhalten kann einen Einfluss haben, wenn sich wie zuvor schon beschrieben, das Verhalten von der GenApi Beschreibung unterscheidet und wenn Register untereinander Abhängigkeiten enthalten oder sich gegenseitig beeinflussen.

- Register können z. B. durch andere Register gesperrt werden, so dass diese zuerst aktiviert oder deaktiviert werden müssen.
- Registerwerte können durch andere Register oder Formeln beeinflusst werden, die z. B. den Maximalwert einschränken.
- Die Kamera kann Registerwerte verändern, wenn andere Register geschrieben werden. Z. B. kann es vorkommen, dass die Kamera automatisch die ROI zurücksetzt, wenn das Pixelformat geschrieben wird.

Herstellerdokumentation beachten

Gleichen Sie bei Problemen das Kameraverhalten genau mit der Herstellerdokumentation ab. Es kann z. B. vorkommen, dass ein Beschreiben von einzelnen ReadWrite Registern von der Kamera fälschlicherweise nicht akzeptiert wird.

GevImageAcquisition-Objekt Verhalten

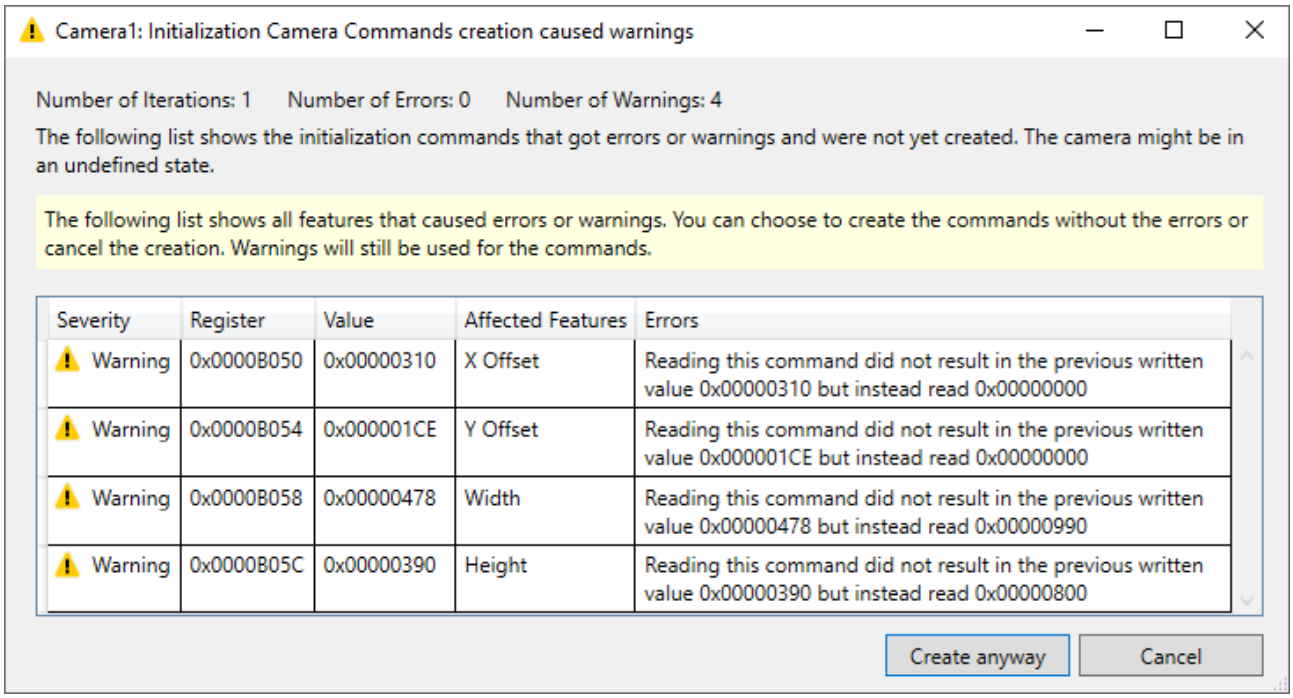
Das GevImageAcquisition-Objekt ist als Teil des TC3 GigE Vision Connector Produkt GigE Vision zertifiziert und verhält sich daher standardkonform. Die Verkettung von Kommandos, sofern von der Kamera unterstützt, und das Iterieren von Selektoren ist in den Einstellungen standardmäßig aktiv. Dadurch werden bei der Kommunikation mit der Kamera alle möglichen Registerkombinationen gelesen oder geschrieben. Dies kann aufgrund der Menge von Kommandos oder des Timings vereinzelt auch zu Problemen führen. Zur Abhilfe können Sie **EnableConcatenatedCommands** im Image Acquisition Module oder das **Iterate Selectors automatically** in den Einstellungen deaktivieren.

Meldungen

Die zuvor beschriebenen Komponenten und deren Verhalten haben unterschiedliche Auswirkungen auf die Erstellung der Initialization Commands bzw. auf die möglichen Meldungen.

Warnungen

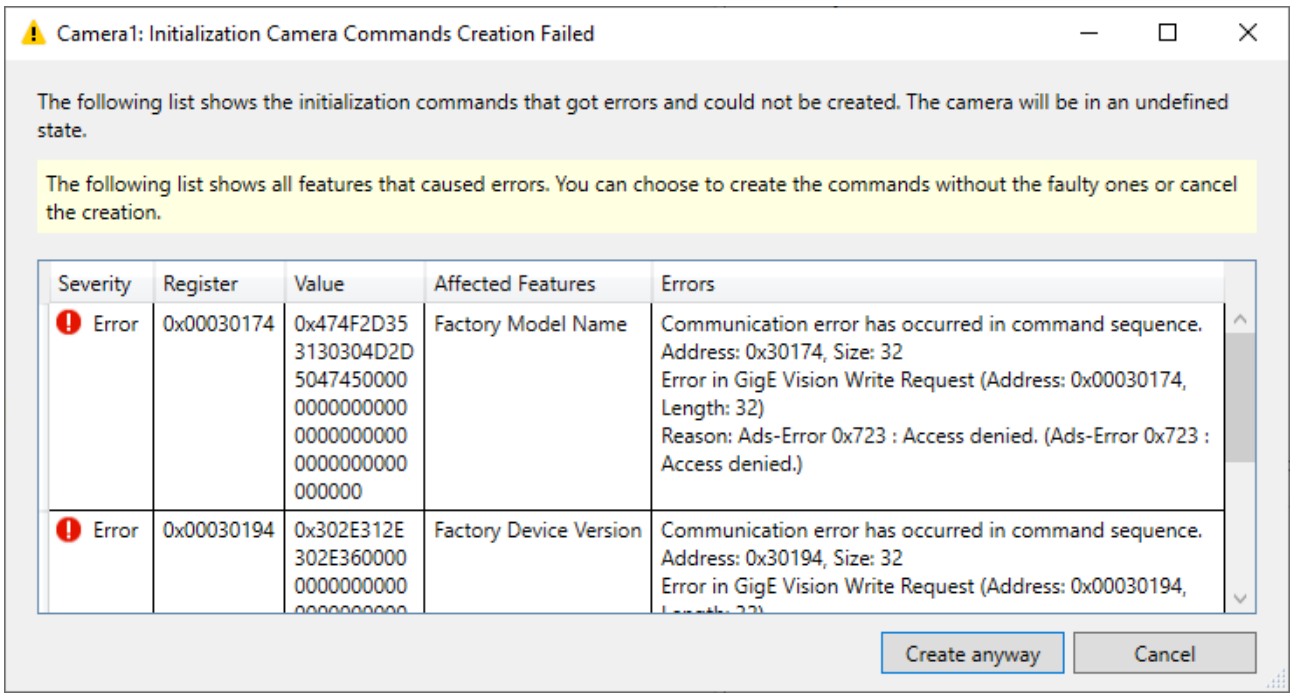
Warnungen werden angezeigt, wenn sich die zurückgelesenen Werte von den zuvor geschriebenen Werten unterscheiden. Die Unterschiede können durch ein abweichendes Kameraverhalten oder Abhängigkeiten zwischen Registern entstehen. Das bedeutet aber nicht, dass die Kamera mit den Initialization Commands nicht funktioniert. Daher haben Warnungen auch keinen Einfluss auf die Erstellung mit **Create anyway**. Im Anschluss sollte getestet werden, ob die gemeldeten Register den richtigen Wert haben und die Kamera sich wie gewünscht verhält.



Ein Beispiel dafür, wann Warnmeldungen auftreten können, ist das Setzen einer ROI. Abhängig von den aktuellen Werten in der Kamera kommt es zu der Warnung, da die Offsets entsprechend der GenApi-Reihenfolge zuerst geschrieben werden. Wenn ein Offset größer 0 geschrieben werden soll, muss die Breite oder Höhe entsprechend kleiner dem Maximalwert sein. In diesem Fall setzt die Kamera dann alle ROI Werte zurück, wodurch der Unterschied entsteht. In diesem Fall kann als Abhilfe die Reihenfolge der ROI-Register in der Konfigurationsliste geändert werden, so dass die Breite und Höhe vor den beiden Offsets geschrieben werden. Falls in Sonderfällen erforderlich kann ein Register auch kopiert, in der Reihenfolge verschoben und so mehrfach geschrieben werden.

Fehler

Fehler werden gemeldet, wenn einzelne Registerwerte nicht geschrieben werden konnten. Dies kann, wie schon zuvor beschrieben, verschiedene Ursachen wie falsche oder zum Zeitpunkt des Schreibens geänderte Grenzwerte, Adressen, Schreibrechte oder Formelberechnungen haben. Die betroffenen Register werden bei der Erstellung mit **Create anyway** weggelassen und in der Konfigurationsliste abgehakt. Im Anschluss sollte getestet werden, ob die weggelassenen Register einen Einfluss auf die Kamerafunktionalität haben.



Falls es sich z. B., wie auf dem Screenshot zu sehen ist, nur um den Modelnamen handelt oder das Register nicht geändert wurde und einen passenden Default-Wert hat, kann die Kamera trotz dieser Fehlermeldung ohne Einschränkungen verwendet werden.

Fatale Fehler

Fatale Fehler treten nur auf, wenn die Kamera innerhalb einer gewissen Zeit nicht antwortet. Dann wird der Prozess abgebrochen und es erscheint eine der folgenden Meldungen. In diesen Fällen muss das entsprechende Timeout angepasst werden.

Timeout

Force IP- oder GVCP-Timeout

Im Zuge des Schreibens der Initialization Commands wird auch der Force IP-Befehl gesendet, sofern dieser konfiguriert ist. Wenn die entsprechende Kamera zum Setzen der IP-Einstellungen bei diesem Befehl sehr lange braucht, kann es zu folgendem Timeout-Fehler kommen:

Abb. 2: Fehler bei Force IP

Persistente IP-Adresse

Wenn kein Force IP verwendet wird, ist es empfehlenswert, eine persistente IP-Adresse einzustellen.

In dem Fall kann das **Force IP** in der Konfigurationsliste [▶ 95] deaktiviert werden. Alternativ dazu oder bei einem GVCP-Timeout kann in den TcCOM-Parametern vom GVCP Module [▶ 108] das `TransmissionTimeout` und/oder die `MaxTimeouts` vergrößert werden. Beachten Sie, dass der `ADS Communication Timeout` im Einstellungsfenster [▶ 91] immer mindestens dem Produkt der beiden vorhergehenden Parameter entsprechen sollte.

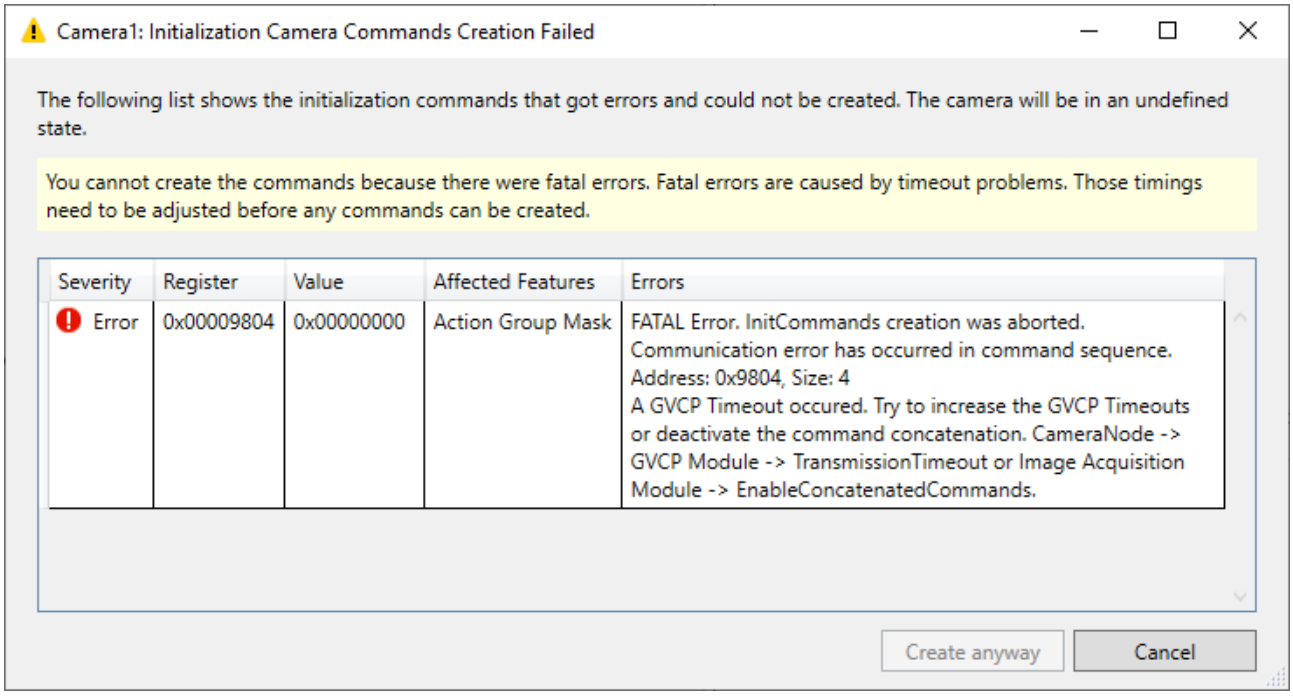


Abb. 3: Kritischer GVCP Timeout Fehler bei Erstellung von Initialization Commands

ADS Timeout

Bei einem ADS Timeout vergrößern Sie den Parameter `ADS Communication Timeout` im Einstellungsfenster [▶ 91].

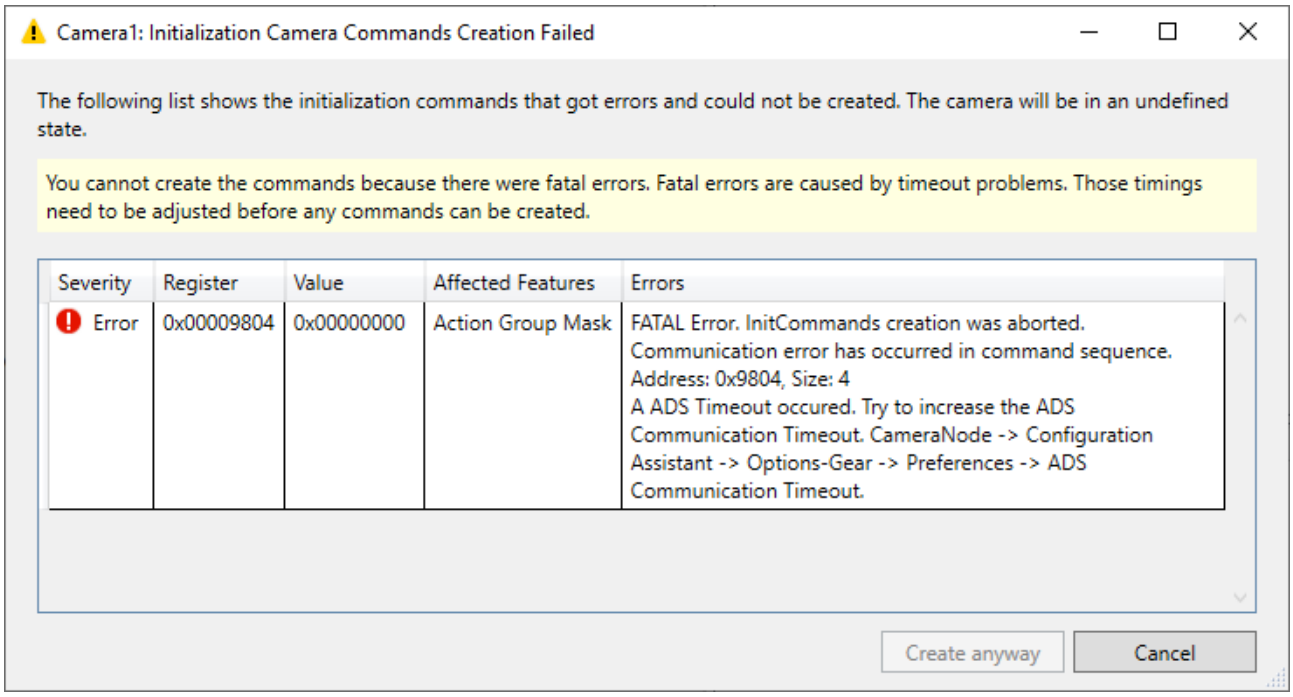


Abb. 4: Kritischer ADS Timeout Fehler bei Erstellung von Initialization Commands

5.4.2.5 TcCOM-Parameter

Folgend sind die Parameter der drei TcCOM-Module Image Acquisition, GVCP und GVSP beschrieben. Die Anpassung dieser Parameter kann bei bestimmten Kameras helfen, eine stabile Verbindung aufzubauen. Zur Veränderung der Parameter sollte der Konfigurationsbaum [▶ 83] genutzt werden.

Zunächst ist hier nur der Parameter `InitializationAutoMode` relevant. Die Einstellung der restlichen Parameter kann allerdings bei Verbindungs- oder Initialisierungsproblemen der Kamera nötig sein.

Neuladen bei Parameteränderungen

Damit Parameteränderungen wirksam werden, müssen die entsprechenden Objekte neu geladen werden.

Image Acquisition Module

Feature	Beschreibung
CameraIPAddress	IP-Adresse der Kamera. Diese IP-Adresse wird bei dem Force-IP-Kommando benutzt. Wenn diese IP-Adresse geändert wird, müssen anschließend die Initialization Commands [▶ 94] neu erstellt werden, damit die Änderung wirksam wird. Zudem muss für das Force-IP-Kommando die entsprechende Option in der Konfiguration [▶ 95] aktiviert sein.
InitializationTimeout	Maximale Zeit zur Übertragung der Initialisierungsparameter auf die Kamera. Abhängig von der verwendeten Kamera kann die Übertragung auch länger dauern, sodass eine Anpassung des Parameters erforderlich ist. Einheit: Millisekunden (ms) Standard: 2.000
InitializationAutoMode	Auswahl, inwiefern die Initialisierung der Kamera von dem ImageAcquisition TcCOM-Objekt automatisch ausgeführt werden soll. Wenn die Kamera nicht initialisiert wurde, kann nicht mit ihr kommuniziert werden. **AUTOINIT_SO**: Die Kamera wird während der Aufstartphase des TcCOM-Objekts initialisiert. In dieser Phase steht nur begrenzt Zeit zu Verfügung. Falls Initialisierungsparameter genutzt werden, kann die Initialisierung auf Grund der potenziellen Timeouts im Normalfall nicht innerhalb dieser Aufstartphase durchgeführt werden. In diesem Fall wird das Verhalten von AUTOINIT_AFTER_SO angewendet, selbst wenn AUTOINIT_SO ausgewählt ist. **AUTOINIT_AFTER_SO**: Das TcCOM-Objekt fährt in den OP-Zustand hoch ohne die Kamera zu initialisieren. Erst wenn aus der SPS die Verbindung zur Kamera geöffnet wird (z. B. mit `FB_VN_SimpleCameraControl.StartAcquisition()` oder mit `FB_VN_GevCameraControl.OpenCamera()`), wird die Kamera automatisch initialisiert. **NO_AUTOINIT**: Die Kamera wird nur dann initialisiert, wenn `GevCameraController.InitializeCamera()` aufgerufen wird. Diese Option ist für den Fall gedacht, dass man die vollständige Kontrolle über die Initialisierung benötigt, um diese zu einem bestimmten Zeitpunkt durchzuführen. Standard: AUTOINIT_AFTER_SO.
EnableConcatenatedCommands	Auswahl, ob beim Schreiben von mehreren Befehlen auf die Kamera, zum Beispiel bei den Initialisierungsparametern, die dazugehörigen Befehle verknüpft oder einzeln versendet werden sollen. Das Verknüpfen der Befehle macht die Übertragung schneller, kann allerdings bei einigen Kameras zu Problemen führen. Standard: aktiviert.

GVCP Module

Feature	Beschreibung
LocalPort	Der lokale UDP Port, über den das GVCP Modul kommuniziert. Ist der Wert null, wird ein automatisch generierter Port verwendet. Standard: 0
MaxTimeouts	Die maximale Anzahl an TransmissionTimeouts, die auftreten dürfen, bevor ein GVCP Paket verworfen wird. Nach jedem Timeout wird das GVCP Paket erneut geschickt. Wenn die maximale Timeout-Anzahl erreicht wird, wird ein Fehler geworfen. Standard: 3 (angepasst an Kamera)
TransmissionTimeout	Nach dem Senden eines GVCP-Pakets wird auf eine Antwort gewartet. Benötigt die Antwort länger als die TransmissionTimeout-Zeit, so wird angenommen, dass das Paket nicht angekommen ist, und es wird ggfs. erneut gesendet. Einheit: Millisekunden (ms) Standard: 600 (angepasst an Kamera) **Der TransmissionTimeout sollte kleiner als die HeartbeatRate sein.**
HeartbeatRate	Der Control Channel der Applikation sendet ein Heartbeat Signal an die Kamera, um zu überprüfen, ob die Verbindung noch gültig ist. Die HeartbeatRate beschreibt die Zeit, nach welcher das nächste Signal gesendet wird. Einheit: Millisekunden (ms) Standard: 1.000 **Die HeartbeatRate sollte nicht beliebig hoch gesetzt werden, da viele Kameras nach einigen Sekunden ohne Heartbeat einen Timeout auslösen.**
DiscoveryTimeout	Das DiscoveryTimeout beschreibt die Zeit, die auf Antworten eines Discovery Kommandos gewartet wird. Das Discovery Kommando wird gesendet, wenn nach Kameras [▶ 72] im Netzwerk gesucht werden soll. Einheit: Millisekunden (ms) Standard: 1.000

GVSP Module

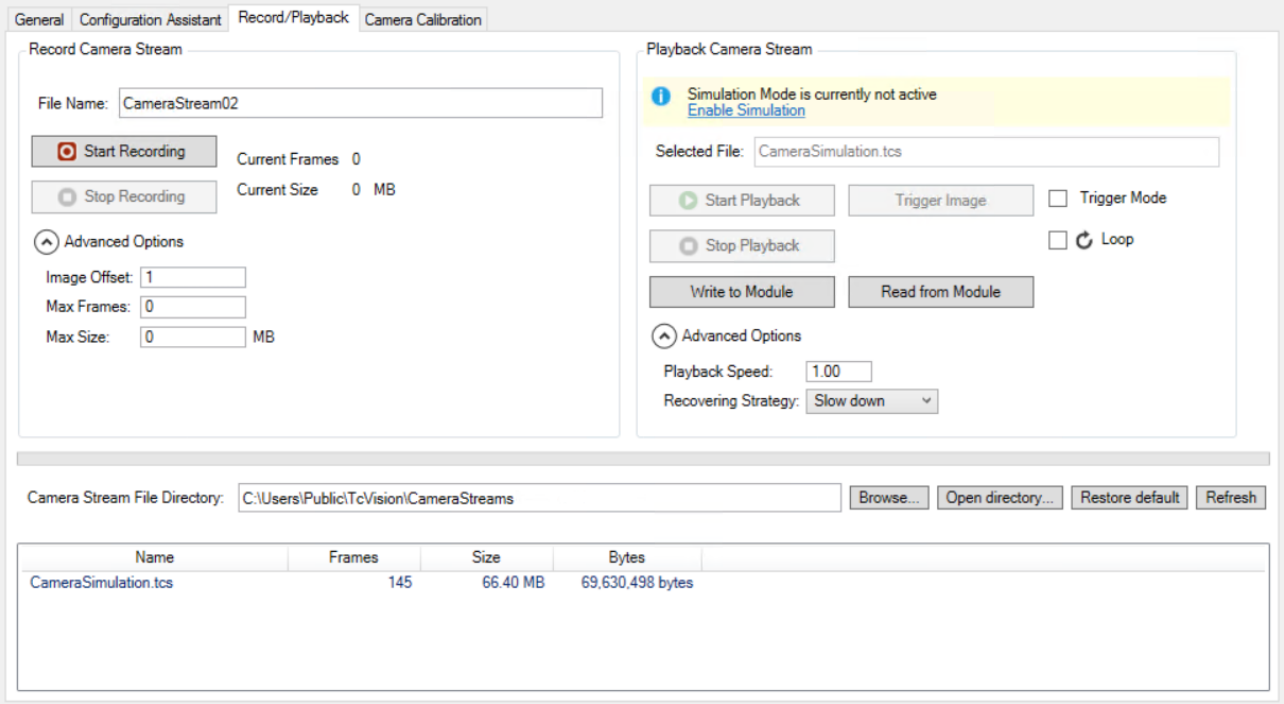
Feature	Beschreibung
LocalPort	Der lokale UDP Port, über den das GVCP Modul kommuniziert. Ist der Wert null, wird ein automatisch generierter Port verwendet. Standard: 0
MaxPacketSize	Die **MaxPacketSize** beschreibt die maximale Paketgröße eines GVSP Paketes. In dieser Größe ist der Header des Ethernet Frames nicht mit inbegriffen. Einheit: Byte Standard: 1.500 **Abhängig vom installierten TwinCAT Realtime Ethernet Treiber und der in den Eigenschaften eingestellten Jumbo Packet Größe, können bis zu 9000 Byte verwendet werden.**
MinPacketDelay	Wird das **MinPacketDelay** erhöht, so werden die GVSP Pakete künstlich verzögert. Dadurch kann die Bandbreite reduziert werden. Einheit: Timestamp counter unit(s) Standard: 0
MaxResendRequests	Die maximale Anzahl an Resend-Requests für ein einzelnes GVSP Paket, welches fehlerhaft war oder zu viel Zeit benötigt hat. Ein Bild wird in einem GVSP Block übertragen, welcher aus mehreren Paketen besteht. Wenn ein Paket eines Blocks nicht erfolgreich übertragen werden konnte, ist das Bild ungültig. Standard: 1
MaxBlockNumber	Maximale Anzahl an GVSP Blöcken, die gleichzeitig bearbeitet werden dürfen. Ist der Wert null, so wird die Anzahl nicht begrenzt. Standard: 5
MaxPacketNumber	Die Maximale Anzahl an GVSP Paketen in einem Block. Blöcke, die mehr Pakete beinhalten, können nicht als gültiges Bild interpretiert werden. Standard: 25.000 **Die Anzahl wird automatisch berechnet, wenn die Option „Automatically calculate the MaxPacketNumber“ in den Preferences Einstellungen aktiv ist.**
BlockTimeout	Das **BlockTimeout** beschreibt die Zeit, die zwischen einzelnen Paketen eines Blocks vergehen darf. Einheit: Mikrosekunden (µs) Standard: 5.000

5.4.3 Record/Playback

In dem Karteireiter **Record/Playback** können Kamera-Streams aufgezeichnet und in die SPS eingespeist werden. Dies kann hilfreich sein, um für die Entwicklung am Schreibtisch einen Kamera-Stream aufzunehmen und diesen später wiederholt offline abspielen zu können. Somit wird das reale Verhalten der Kamera-Hardware mit den Original-Bildern simuliert.

Entweder ist der Simulationsmodus aktiviert, dann können Kamera-Streams abgespielt werden, oder der Simulationsmodus ist deaktiviert, dann können Kamera-Streams aufgenommen werden. Beides gleichzeitig ist nicht möglich.

Ein Kamera-Stream besteht aus Einzelbildern, die zusammen mit korrespondierenden Zeit-Markern in einem eigenen Format (.tcs) abgespeichert sind. Durch das zusätzliche Speichern der Zeit-Marker kann das zeitliche Verhalten der Kamera an der Maschine anwendungsnah simuliert werden. Die Bilddaten werden in genau dem Format gespeichert, in dem sie von der Kamera geschickt werden; es geschieht keine weitere Konvertierung.



Kameravoraussetzung

Die Kamera muss Zeitstempel unterstützen und valide Werte in den Bild-Header eintragen. Die Zeitstempel werden pro Bild benötigt, um das zeitliche Verhalten simulieren zu können.

Systemvoraussetzungen

Setzen Sie für das Aufzeichnen und Abspielen von Kamera-Streams schnelle SSD-Festplatten und min. 8 GB Arbeitsspeicher ein. Wenn langsame Speichermedien verwendet werden, reduzieren Sie ggfs. die Bildrate über den Parameter **Image Offset**.

(De)aktivieren von TcCOM-Objekten

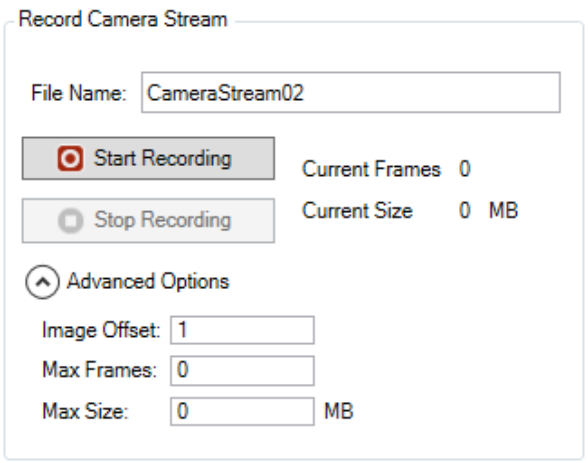
Das Aktivieren / Deaktivieren der Kamera-TcCOM-Objekte sollte nicht manuell erfolgen, da beim Wechsel vom Simulationsmodus und von der Ads Communication im Hintergrund weitere Verknüpfungen angepasst werden müssen.

Die drei Bereiche des Record/Playback-Assistenten (Record, Playback, Stream-Dateien) sind folgend erklärt.

Aufzeichnen eines Kamera-Streams:

Das Aufzeichnen eines Kamera-Streams ist nur mit einer aktiven Kameraverbindung möglich. Entsprechend muss der Simulationsmodus der Kamerainstanz [▶ 72] deaktiviert und die Standardakquisition ausgewählt sein. Zusätzlich wird das Ads Communicator TcCOM-Modul benötigt.

Im Bereich **Record Camera Stream** befinden sich die Einstellungen zum Aufzeichnen eines Streams:



File Name	Textfeld für den Dateinamen des Kamera-Streams. Der Pfad, in dem die Datei abgelegt wird, wird in Auswahl von Dateipfad und Streams [▶ 115] festgelegt.
Start Recording	Button zum Starten einer Aufzeichnung. Nur aktiv, wenn eine Aufzeichnung möglich ist.
Stop Recording	Button zum Stoppen einer Aufzeichnung. Nur während einer Aufzeichnung aktiv.
Current Frames	Anzahl der Bilder im aktuell aufgenommenen Kamera-Stream.
Current Size	Dateigröße des aktuell aufgenommenen Kamera-Streams.
Advanced Options	Option zum Anzeigen und Verstecken von erweiterten Optionen.
Image Offset	Angabe jedes wievielte Bild aufgezeichnet wird. Abhängig von der Schreibgeschwindigkeit des Speichermediums, der Bildgröße und der Bilder pro Sekunde kann ein Auslassen von Bildern erforderlich sein. Standard: 1 (d. h. jedes Bild wird aufzuzeichnen versucht).
Max Frames	Maximal erlaubte Anzahl an Bildern im aufzunehmenden Kamera-Stream. Sobald diese Anzahl an Bildern erreicht wird, wird die Aufnahme automatisch beendet. Standard: 0 (d. h. keine Grenze).
Max Size	Maximal erlaubte Dateigröße des aufzunehmenden Kamera-Streams in MB. Sobald diese Dateigröße erreicht ist, wird die Aufnahme automatisch beendet. Standard: 0 (d. h. keine Grenze).

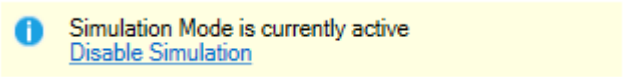
Genug Speicherplatz frei?

Stellen Sie vor der Aufzeichnung eines Streams sicher, dass genügend Speicherplatz auf Ihrem Zielsystem frei ist oder limitieren Sie die Stream-Größe ggfs. mit **Max Frames** und **Max Size**.

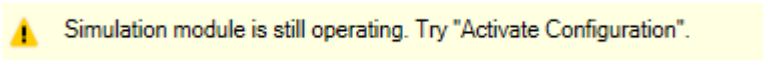
Folgende Bedingungen müssen erfüllt sein, damit die Aufzeichnung eines Streams möglich ist:

- Der Simulationsmodus ist deaktiviert.
- Ein Ads Communication Objekt existiert und ist aktiviert.
- TwinCAT ist im Runmode.
- Die Kamera muss sich im `ACQUIRING` Status befinden.

Falls diese Bedingungen nicht erfüllt sind, können zudem folgende Hinweise und Lösungsvorschläge erscheinen:

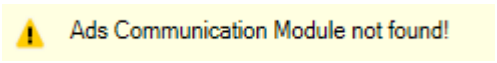


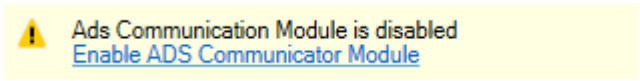
- Unter „Simulation Mode“ wird der Zustand verstanden, in dem ein Stream abgespielt, aber nicht aufgezeichnet werden kann. Über den Link **Disable Simulation** im Hinweis kann der Zustand geändert werden. Dies setzt zusätzlich voraus, dass die Konfiguration erneut aktiviert wird .



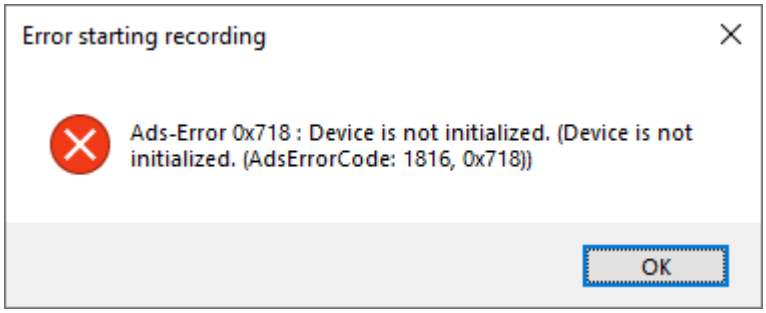
- Der Simulation Mode wurde ausgeschaltet, die Änderung wird aber erst nach einem Aktivieren der Konfiguration übernommen.

- Das Aufzeichnen eines Kamera-Streams ist nur im TwinCAT Run Mode möglich. In allen anderen TwinCAT Zuständen wird diese Warnung angezeigt.





- Das Aufzeichnen eines Kamera-Streams ist nur möglich, wenn eine offene Verbindung zur Kamera besteht, da vor der Aufzeichnung Daten aus der Kamera gelesen werden. Daher sollte sich die Kamera auch zu diesem Zeitpunkt schon im `ACQUIRING` Status befinden.



- Zum Aufzeichnen von Kamera-Streams wird das Ads Communication Module benötigt. Über den Link im Hinweis kann diese aktiviert werden. Falls noch kein ADS Communicator Objekt existiert, kann es über **Add New Item…** auf dem Kamera-Objekt und **Try to fix** im General [▶ 74] Tab hinzufügt werden.

  Die Änderung wird nach dem Aktivieren der Konfiguration übernommen.

ADS Communicator Modul im Vorhinein aktivieren

Falls Sie planen, zu einem späteren Zeitpunkt einen Kamera-Stream aufzuzeichnen, ohne vorher TwinCAT zu stoppen (z. B. während der Laufzeit einer Maschine), müssen Sie das ADS Communicator Modul im Vorhinein aktivieren.

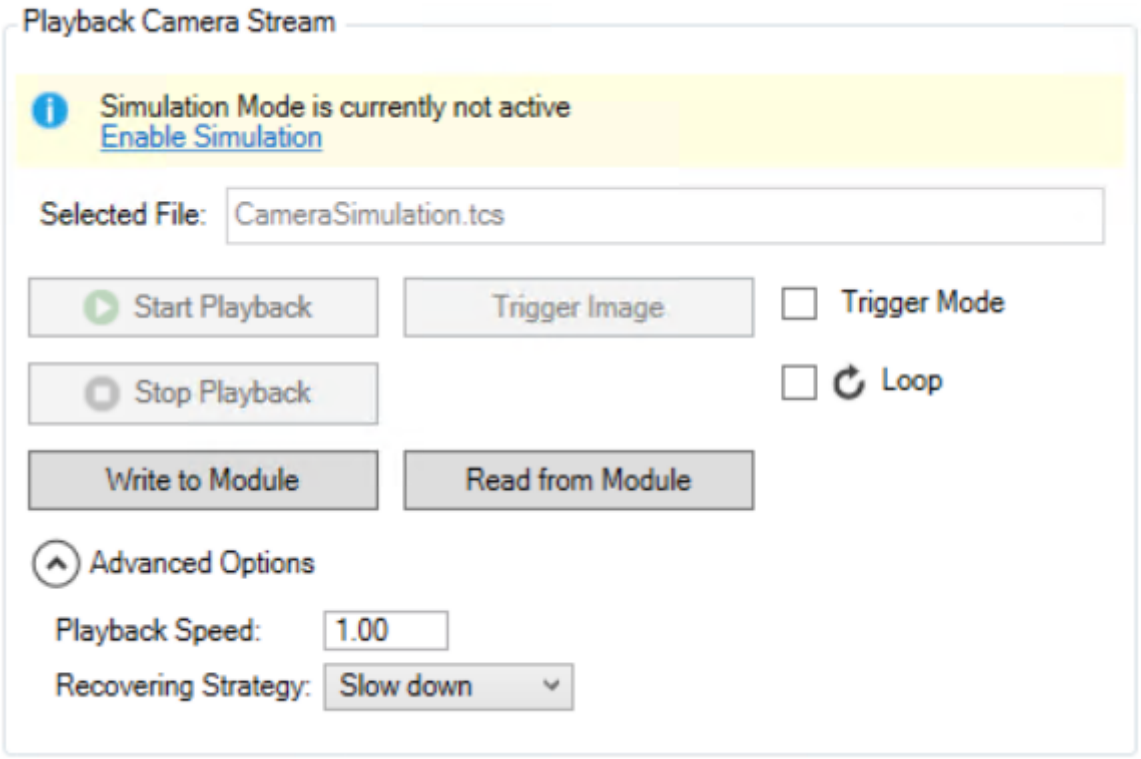
Wenn das Ads Communicator Objekt aktiviert ist, kann es vorkommen, dass Bilder von der Kamera für kurze Zeit weiterhin intern im Objekt referenziert sind und daher nicht direkt in anzeigbare Bilder transformiert werden können. Für weitere Details siehe Anzeigbare Bilder [▶ 161].

Abspielen eines Kamera-Streams:

Zum Abspielen eines Kamera-Streams muss der Simulationsmodus der Kamera aktiviert werden; dies ist entweder über den General [▶ 74] Tab des Kamera-Objekts oder in der unten erwähnten Meldung möglich.

Wenn Sie die Konfiguration mit Write to Module ins TcCom Objekt schreiben und im Projekt speichern, sind diese Werte nach der Aktivierung des Projekts bereits vorhanden und die Simulation beginnt direkt mit dem Start Acquisition Kommando aus der SPS. Das hat den Vorteil, dass Sie nicht jedes Mal händisch die Einstellungen neu setzen und den Stream starten müssen.

Im Bereich **Playback Camera Stream** befinden sich die Einstellungen zum Abspielen eines Kamera-Streams:

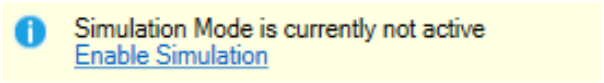

Playback Camera Stream
Simulation Mode is currently not active
Enable Simulation
Selected File:
CameraSimulation.tcs
Start Playback
Trigger Image
Trigger Mode
Stop Playback
Loop
Write to Module
Read from Module
Advanced Options
Playback Speed:
1.00
Recovering Strategy:
Slow down

Selected File	Textfeld zum Anzeigen der ausgewählten Stream-Datei. Die Auswahl erfolgt über die Liste der vorhandenen Streams.
Start Playback	Button zum Starten der Abspielung. Nur freigegeben, wenn das Abspielen möglich ist. Das Playback beginnt immer bei Bild 1, unabhängig davon, wo der Stream ggfs. vorher gestoppt wurde.
Stop Playback	Button zum Stoppen der Abspielung. Nur freigegeben, während ein Stream abgespielt wird.
Trigger Mode	Auswahl des Trigger Modes, in dem die Frames des Streams einzeln getriggert werden können.
Trigger Image	Button zum Triggern eines Bildes. Nur aktiv, wenn der Trigger Mode ausgewählt ist und während ein Stream abgespielt wird.
Loop	Auswahl, ob der Stream einmalig oder in einer Endlosschleife (Loop) abgespielt werden soll.
Write to Module	Schreibt die aktuellen Einstellungen in das TcCom Modul. Gespeichert werden: Playback Mode, Playback Speed, Playback Strategy, Loop und der Pfad der ausgewählten Stream-Datei.
Read from Module	Liest die gespeicherten Werte aus dem Modul und setzt diese Einstellungen.
Advanced Options	Ein- und Ausblenden von fortgeschrittenen Einstellungen.
Playback Speed	Beschleunigungs-/Verzögerungsfaktor zur Abspielgeschwindigkeit. (>1 Beschleunigung; <1: Verzögerung) Standard: 1.00 = Geschwindigkeit entsprechend der Aufnahme
Recovering Strategy	Strategievorgabe für den Fall, dass das Abspielen mit der vorgegebenen Geschwindigkeit nicht möglich ist: • Slow down: Verzögerung der Abspielgeschwindigkeit (Standardeinstellung) • Skip images: Überspringen von Bildern • Abort on lag: Abbruch der Abspielung (nicht empfohlen)

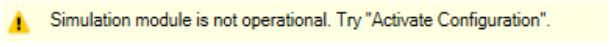
Folgende Bedingungen müssen erfüllt sein, damit das Abspielen eines Streams möglich ist:

- Der Simulationsmodus ist aktiviert.
- TwinCAT ist im Runmode.

Oberhalb des Textfeldes mit dem ausgewählten Dateinamen, können zudem folgende Hinweise und Warnungen erscheinen:



- Unter dem Simulation Mode wird der Zustand verstanden, unter dem ein Stream abgespielt werden kann. Über den Link **Enable Simulation** sowie einem anschließenden Aktivieren der Konfiguration kann dieser Zustand erreicht werden.

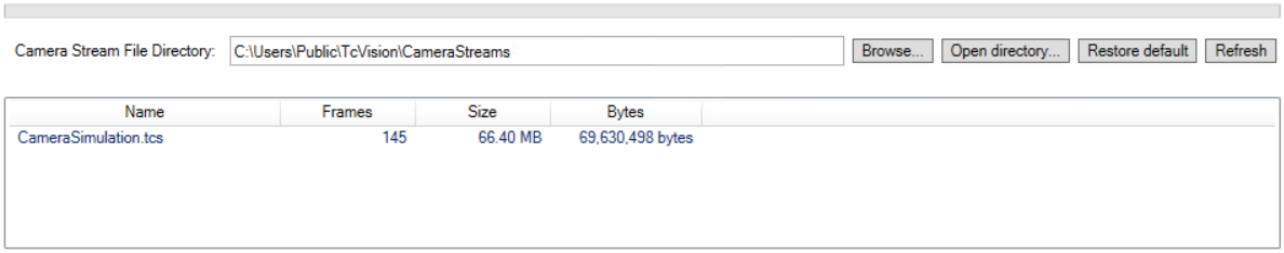


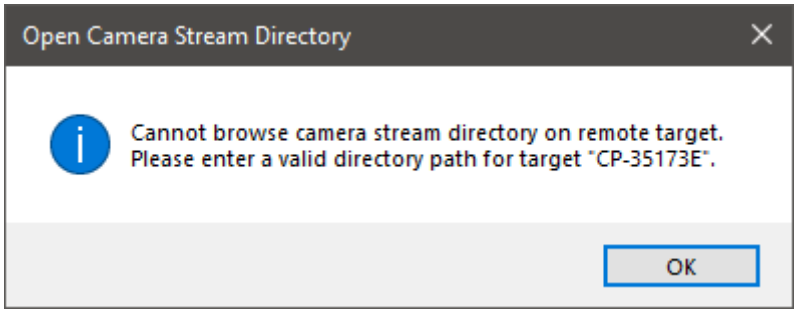
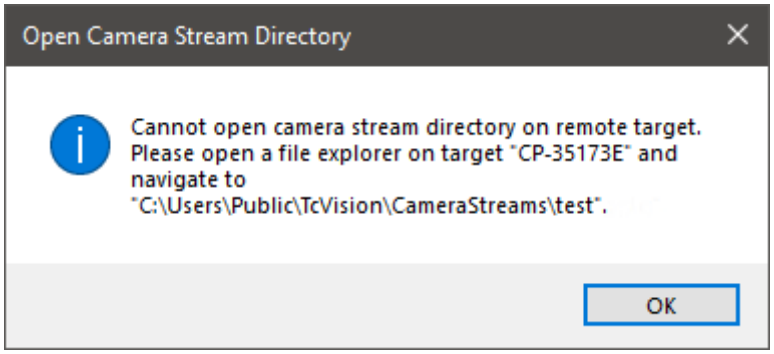
- Der Simulation Mode wurde aktiviert, die Änderung wird jedoch erst nach dem Aktivieren der Konfiguration aktiv.

- Das Abspielen eines Kamera-Streams ist nur im TwinCAT Run Mode möglich. In allen anderen TwinCAT Zuständen wird diese Warnung angezeigt.

Auswahl von Dateipfad und Streams

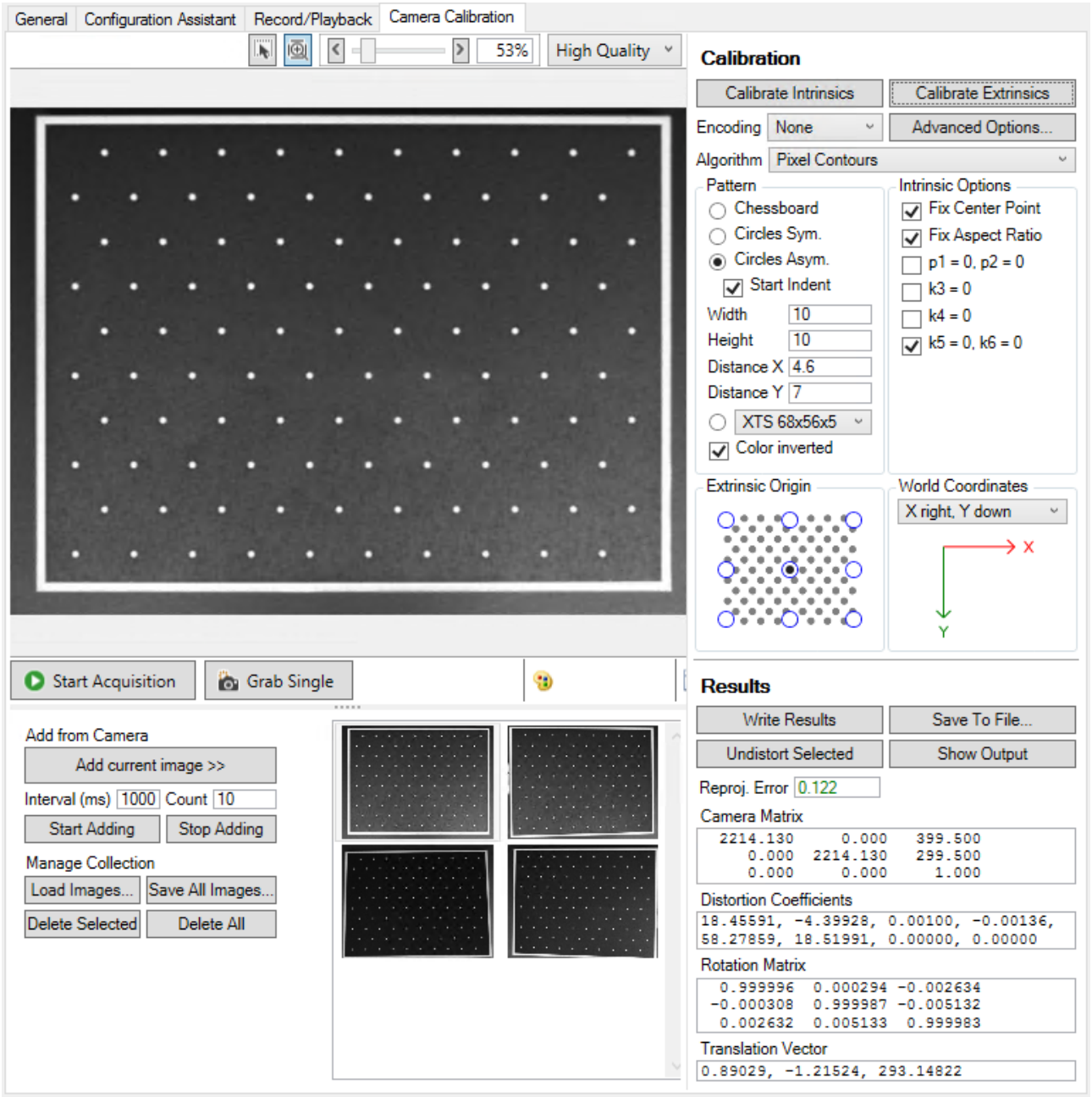
Unten befindet sich die Dateipfad- und Stream-Auswahl:



	Status-Leiste für Aufzeichnung oder Wiedergabe
Camera Stream File Directory	Aktuell eingestellter Dateipfad. Dieser Pfad wird sowohl zum Lesen von bereits aufgezeichneten Streams als auch zum Schreiben von neuen Streams benutzt. Es werden ebenfalls Unterordner nach Stream-Dateien durchsucht. **Der Dateipfad bezieht sich immer auf das Zielsystem und nicht auf das Entwicklungssystem.** **Stellen Sie sicher, dass dieser Dateipfad nicht durch Write-Filter o. Ä. beeinträchtigt wird.** Siehe Wichtige Pfade [▶ 2947].
Browse…	Button öffnet ein Auswahlfenster zum Browsen des Zielordners. **Dies ist nur möglich, wenn das Entwicklungssystem gleichzeitig auch das Zielsystem ist.** Ansonsten erscheint folgende Meldung: 
Open directory…	Button zum Öffnen des angegebenen Dateipfads im lokalen Datei-Explorer. Dies ist ebenfalls durch einen Doppelklick auf den freien Bereich in der Dateiliste zu erreichen. **Dies ist nur möglich, wenn das Entwicklungssystem gleichzeitig auch das Zielsystem ist.** Ansonsten erscheint folgende Meldung: 
Restore default	Button zum Setzen des Standard-Dateipfads.
Refresh	Button zum aktualisiert der Stream Liste.
Name	Liste aller Kamera-Streams im ausgewählten Dateipfad, über die auch die Stream-Auswahl zum Abspielen erfolgt.
Frames	Anzahl der Bilder, aus denen der jeweilige Kamera-Stream besteht.
Size	Dateigröße des jeweiligen Kamera-Streams in MB.
Bytes	Die Größenangabe in Anzahl an Bytes.

5.4.4 Camera Calibration

In dem Tab Camera Calibration kann eine geometrische Kalibrierung von Flächenkameras mit Hilfe eines Kalibriermusters vorgenommen werden.



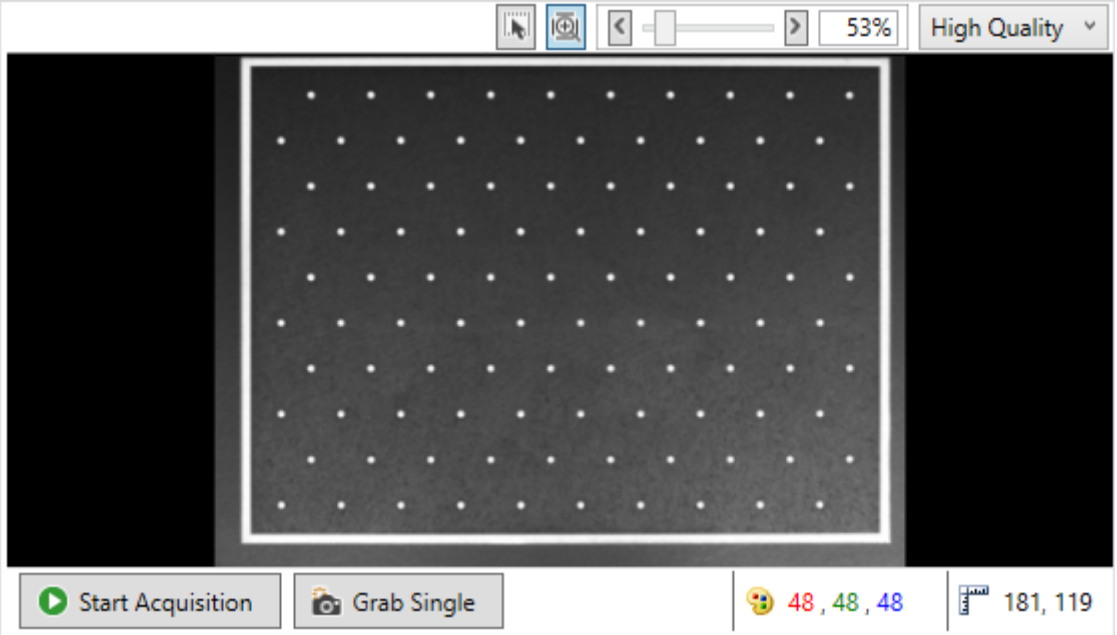
Bei der geometrischen Kamerakalibrierung von Flächenkameras wird der Abbildungsvorgang vom Bild aus rekonstruiert. Hierzu werden Punkte im Bild benötigt deren Position in der realen Welt bekannt sind. Dazu gibt es Kalibriermuster mit definierten Punkten von denen Aufnahmen gemacht werden.

Unterstützte Kalibriermuster:

- Schachbrettmuster [▶ 126]
- Symmetrische Kreismuster [▶ 127]
- Asymmetrische Kreismuster [▶ 128]
- Individuelle Kreismuster [▶ 129]

Bildaufnahme / -auswahl

Für die Bildaufnahme gibt es wie im Configuration Assistant [▶ 79] Vorschaubild, bei dem sich oberhalb Anzeigeoptionen und unterhalb Controls zur Bildaufnahme befinden:



Darstellung des Vorschaubildes

Das Bild wird immer so dargestellt, wie es von der Kamera empfangen wird. Ein Bild im Bayer-Format wird also unkodiert als Grauwertbild angezeigt. An dieser Stelle wird das eingestellte Encoding noch nicht auf das Bild angewandt. Das Encoding ist nur für den Kalibrierungsprozess, aber nicht für das Vorschaubild relevant.

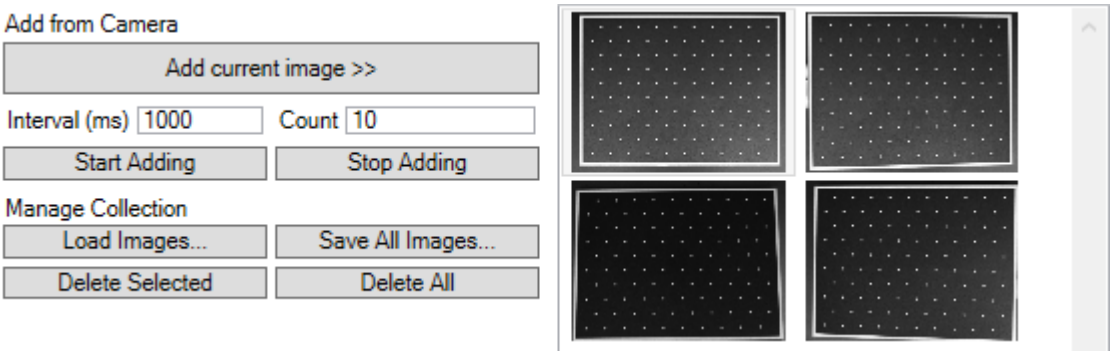
Oberhalb des Vorschaufensters:

- **Interpolation des Vorschaubildes:**
  Combobox zum Auswählen ob eine High Quality oder Nearest Neighbor Interpolation des Vorschaubildes erfolgen soll. Die Auswahl betrifft nur die Vorschau, die Kalibrierung erfolgt anhand der original Bilddaten.
- **Größe des Vorschaubildes**
  Ist **Fit to window** ausgewählt, wird das Bild jeweils auf die Größe des Vorschaufensters skaliert. Alternativ kann über den Slider auch eine Anzeigegröße vorgegeben werden.

Unterhalb des Vorschaufensters:

- **Start Acquisition / Stop Acquisition**
  Button zum Starten bzw. Stoppen der Bildaufnahme. Der Streaming-Mode der Kamera muss dazu aktiv sein.
- **Grab Single**
  Button zum Triggern eines einzelnen Bildes.
- **Pixel Value**
  Hier wird der Farbwert des Pixels an der Maus-Cursor-Position auf dem Bild wiedergegeben.
- **Cursor Position**
  Hier wird die Maus-Cursor Position auf dem Bild wiedergegeben. Dabei ist zu beachten, dass der Null-Punkt bei Bildern immer oben links ist.

Befindet sich das Kalibriermuster passend im Sichtfeld der Kamera, kann das dazu gehörige Bild für die Kalibrierung ausgewählt werden. Hierzu befinden sich links unterhalb des Vorschaubildes die Controls und rechts daneben die Sammlung der Bilder für die Kalibrierung.

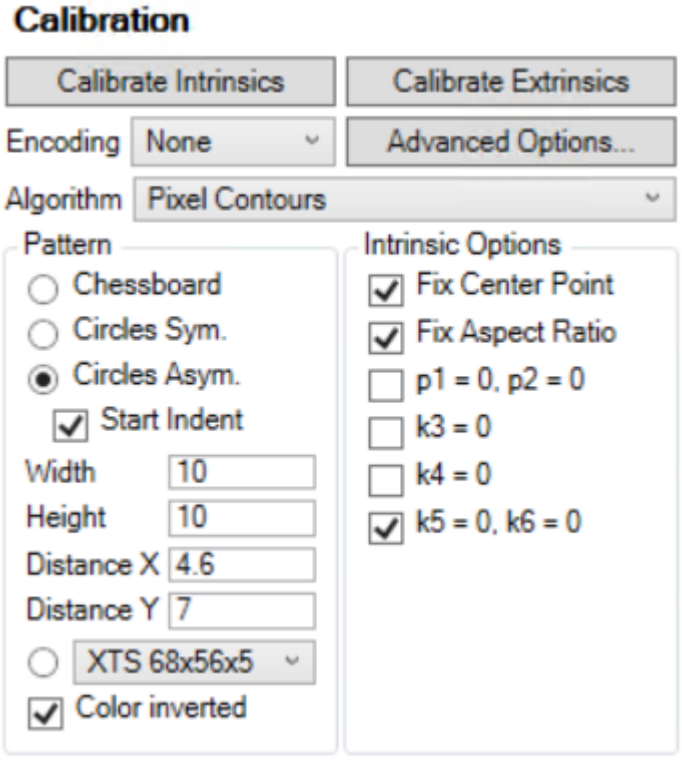


- **Add current image >>**
  Über **Add current image** wird das aktuell angezeigte Kamerabild der Sammlung für die Kalibrierung hinzugefügt.
- **Start / Stop Adding**
  Anstelle eines einzelnen Bildes kann auch eine Bildserie der Sammlung hinzugefügt werden. Die Bildserie wird über **Start Adding** gestartet. Anschließend werden im Abstand der bei **Interval** eingestellten Zeit Bilder hinzugefügt bis die bei **Count** eingetragene Anzahl erreicht ist oder das Hinzufügen von Bildern über **Stop Adding** beendet wird.
- **Load Images...**
  Alternativ zur Bildaufnahme können über **Load Images** gespeicherte Bilder geladen werden. Dabei ist zu beachten, dass diese mit demselben Kamerasystem, mit denselben Kameraeinstellungen, denselben Objektiveinstellungen und demselben Kalibriermuster aufgenommen wurden.
- **Save All Images...**
  Über **Save All Images** kann die vollständige Bildsammlung gespeichert werden.
- **Delete Selected**
  Über **Delete Selected** werden die in der Sammlung markierten Bilder gelöscht.
- **Delete All**
  Über **Delete All** werden alle Bilder der Sammlung gelöscht.

Bildformat

An dieser Stelle dürfen nur Mono8, RGB24 und Bayer8 Bilder verwendet werden.

Einstellungen und Berechnungen



Encoding

Für Vermessungsaufgaben empfiehlt es sich, eine monochrome Kamera zu verwenden. Ist es dennoch notwendig eine Farbkamera mit Bayer-Pattern zu kalibrieren, muss das entsprechende Bayer-Pattern bei **Encoding** angegeben sein. Dies wird standardmäßig automatisch ausgewählt.

Algorithm

Für die Kreismustererkennung kann zwischen verschiedenen Funktionen der Kantenlokalisierung gewählt werden. Durch die Subpixel Funktion lässt sich, je nach Bildqualität, das Ergebnis verbessern. Damit benötigt der gesamte Vorgang aber auch mehr Zeit. Weitere Informationen zu der Subpixel Funktionalität finden Sie im Measurement [▶ 1511] Kapitel.

- **Pixel Contours**
  Kontourbasiert
- **Subpixel Interpolation**
  Mit Subpixel durch Interpolation
- **Subpixel Erf Approximation**
  Mit Subpixel nach der Erf Approximation

Definition des Kalibriermusters

Unter **Pattern** wird das verwendete Kalibriermuster ausgewählt und definiert:

- **Chessboard**
  Verwendung eines Schachbrettmusters
- **Circles Sym.**
  Verwendung eines symmetrischen Kreismusters
- **Circles Asym.**
  Verwendung eines asymmetrischen Kreismusters
  - **Start Ident**
    Checkbox ob die erste Zeile des Kreismusters eingerückt ist. Im Standard ohne Haken wird eine nicht eingerückte Zeile erwartet.
- **Width**
  Anzahl der Merkmalspunkte in horizontaler Richtung

- **Height**
  Anzahl der Merkmalspunkte in vertikaler Richtung
- **Distance X**
  Abstand im Weltkoordinatensystem der Merkmalspunkte in horizontaler Richtung
- **Distance Y**
  Abstand im Weltkoordinatensystem der Merkmalspunkte in vertikaler Richtung
- **Dropdown-Menü für individuelle Kreismuster**
  Auswahl eines individuellen Kreismusters (Einträge werden nur angezeigt, wenn entsprechende Dateien in dem vorgesehenen Ordner gefunden würden)
- **Color inverted**
  Checkbox ob die Farbe des Bildes invertiert werden soll. Im Standard ohne Haken werden schwarze Objekte auf weißem Hintergrund erwartet.

Symmetrische Muster

Bei gänzlich symmetrischen Mustern kann die Zuordnung der gefundenen zu den definierten Punkten abweichen bzw. kann der Koordinatenursprungspunkt in einem anderen Quadranten liegen. Wenn sich das Muster auf dem ausgewählten Bild in einer anderen Orientierung befindet als definiert wurde, kann das ebenfalls vorkommen.

Daher sollten sie nach abgeschlossenem Kalibiervorgang kontrollieren, ob das Koordinatensystem ihrer Vorgabe entspricht.

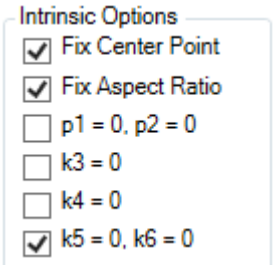
Prüfung auf gefüllte Kreise

Bei Kreismustern wird zusätzlich überprüft, ob die Kreise in der entsprechenden Farbe gefüllt sind. Treten an dieser Stelle Abweichungen auf, wird der entsprechende Punkt nicht als Teil des Musters erkannt.

Hinweise zu den Mustern und den Einstellungen der Parameter sind bei den unterstützten Kalibriermustertypen zu finden:

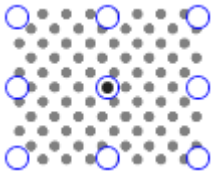
- Schachbrettmuster [▶ 126]
- Symmetrische Kreismuster [▶ 127]
- Asymmetrische Kreismuster [▶ 128]
- Individuelle Kreismuster [▶ 129]

Intrinsic Options



- **Fix Center Point:** Der Prinzipalpunkt entspricht dem Bildmittelpunkt.
- **Fix Aspect Ratio:** Es werden quadratische Pixel auf dem Kamerasensor angenommen, sodass das Seitenverhältnis der Pixel ist in x- und y-Richtung gleich ist.
- **p1=0, p2=0:** Es wird angenommen, dass keine tangentialen Verzeichnungen existieren und die entsprechenden Koeffizienten werden auf 0 gesetzt.
- **k3=0:** nimmt den Radialverzerrungskoeffizient k_3 als 0 an
- **k4=0:** nimmt den Radialverzerrungskoeffizient k_4 als 0 an
- **k5=0, k6=0:** nimmt die Radialverzerrungskoeffizienten k_5 und k_6 als 0 an

Extrinsic Origin



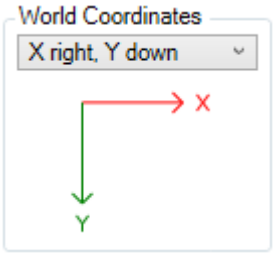
Extrinsic Origin legt den Ursprungspunkt des Weltkoordinatensystems im Kalibriermuster fest.

Je nach ausgewähltem Kalibriermuster wird ein entsprechendes Hintergrundbild angezeigt.

Ursprungspunkt

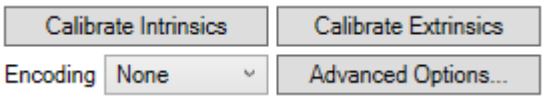
Der ausgewählte Ursprungspunkt bezieht sich auf den entsprechenden Punkt des Kalibriermusters, nicht auf das Gesamtbild.

World Coordinates



World Coordinates legt die Ausrichtung des Weltkoordinatensystems fest. Damit kann eine abweichende Ausrichtung im Vergleich zum Bildkoordinatensystem eingestellt werden. Bei der abgebildeten Standardeinstellung stimmen die beiden Koordinatensysteme überein.

Aktionen



Calibrate Intrinsics

Durch **Calibrate Intrinsic**s erfolgt die Berechnung der Kameramatrix und der Verzeichnungskoeffizienten. Dazu werden alle Bilder der Auswahl verwendet. Diese müssen dazu alle vollständig das definierte Kalibriermuster enthalten. Ist dies nicht der Fall, ist dies an einem negativen **Reproj. Error** zu erkennen.

Calibrate Extrinsics

Durch **Calibrate Extrinsics** erfolgt die Berechnung der Rotationsmatrix und des Translationsvektors. Voraussetzung dazu ist die intrinsische Kalibrierung (Kameramatrix und Verzeichnungskoeffizienten) sowie die Auswahl des Bildes, in dem sich die Punkte des 2D-Kalibriermusters in der späteren Messebene befinden.

Dauer einer Kalibrierung

Die Dauer einer Kalibrierung erhöht sich mit der Anzahl der Referenzpunkte im Kalibriermuster und der Anzahl an Kalibrierbildern. Achten Sie daher bei der Auswahl eines Kalibriermusters darauf, dass die Dauer in einem für Sie akzeptablen Rahmen bleibt.

Abweichung der Ergebnisse

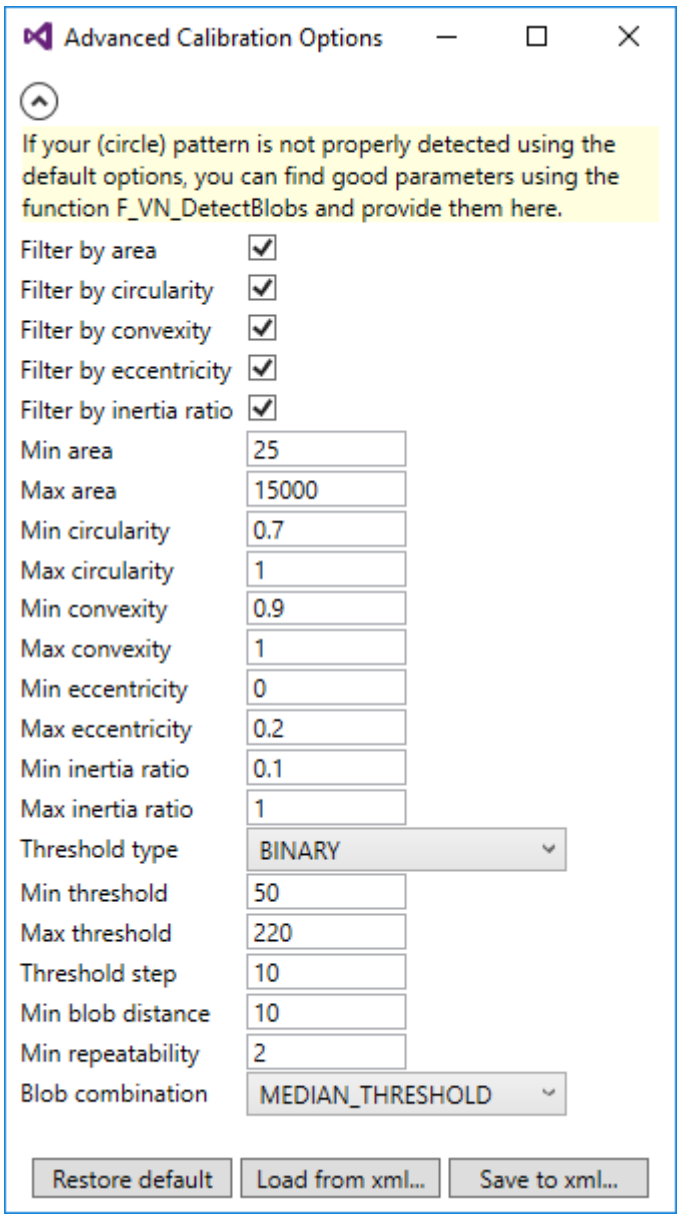
Aufgrund des verwendeten Algorithmus können die Ergebniswerte der Kalibrierung von Aufruf zu Aufruf leicht im Nachkommastellenbereich voneinander abweichen. Das betrifft sowohl die Ergebnisse der Kalibrierung im Kalibrationsassistenten und die in der SPS als auch zwischen beiden Kalibrationsvarianten.

Übernahme der Einstellungen in das TwinCAT Projekt

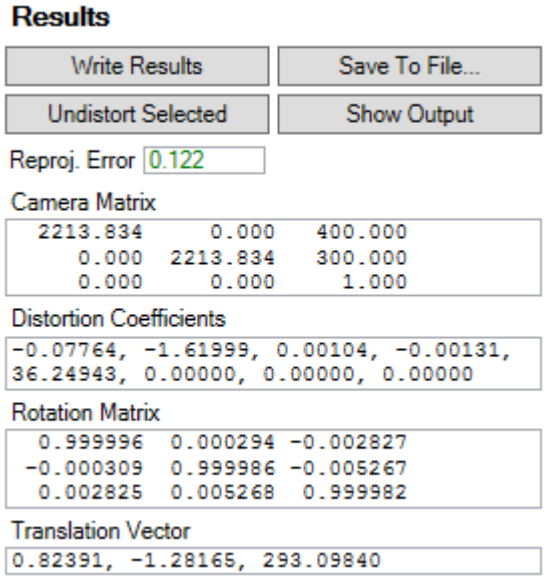
Die eingestellten Werte und Optionen werden nur nach dem Ausführen der **Calibrate Intrinsics**- oder **Calibrate Extrinsics**-Funktion in das TwinCAT Projekt übernommen. Wenn das Projekt danach abgespeichert wird, werden diese Werte beim erneuten Öffnen wieder geladen. Ansonsten werden immer die Default-Einstellungen verwendet.

Advanced Options

Die Kreismustererkennung kann über die **Advanced Calibration Options** nachparametriert werden. Dies sollte nur erfolgen, wenn es Schwierigkeiten mit den Standardeinstellungen gibt. Zum besseren Verständnis der Einstellparameter hilft ein Blick in die Parametrierung der Funktion F_VN_DetectBlobs [▶ 1207].



Ergebnisse



Reproj. Error
Der Reprojection Error gibt Aufschluss über das Ergebnis der intrinsischen Kalibrierung:

- **>0**: gültiges Ergebnis
  - **<1**: gutes Ergebnis
  - **>2**: ungenaues Ergebnis
- **<0**: Kein Ergebnis und spezifische Fehlernummer:
  - **-100**: kein Bild gefunden
    Für die intrinsische Kalibrierung wird auf die Sammlung an Bildern unterhalb des Vorschaufensters zurückgegriffen. Das Livebild einer Kamera wird nicht berücksichtigt.
  - **-200**: das Muster konnte in mindestens einem Bild nicht gefunden werden.
    Auf allen Bildern in der Sammlung unterhalb des Vorschaufensters muss das Muster vollständig inkl. aller Merkmalspunkte abgebildet sein. Zur besseren Analyse werden unter *C:\ProgramData\Beckhoff\Vision_CalibrationAssistantOutput* Ergebnisbilder erstellt, die zeigen, welche Merkmalspunkte gefunden wurden.
  - **-300**: keines der vorhandenen Bilder enthält das gewählte Muster.
    Siehe Beschreibung zu -200, zudem empfiehlt es sich die Konfiguration des Musters zu überprüfen.
  - **-400**: Die benutzerdefinierte Beschreibungsdatei für ein individuelles Kalibriermuster ist ungültig.
  - **-500**: Es ist ein Fehler aufgetreten.

Camera Matrix

Die Kameramatrix beschreibt den Abbildungsvorgang von 3D zu 2D.

$$CameraMatrix = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

- Die Parameter $\mathbf{f}_x$ und $\mathbf{f}_y$ geben die Brennweite in x- und y-Richtung wieder.
- Die Parameter $\mathbf{c}_x$ und $\mathbf{c}_y$ beschreiben den Schnittpunkt der optischen Achse mit dem Bild, der als Hauptpunkt bezeichnet wird. Im Idealfall liegt dieser im Bildmittelpunkt. Ist bei den **Intrinisic Options** der **Fix Center Point** ausgewählt, werden die Parameter $\mathbf{c}_x$ und $\mathbf{c}_y$ automatisch auf den Bildmittelpunkt gelegt.

Distortion Coefficients

Durch Abbildungsfehler des optischen Systems kommt es u.a. zu Verzeichnungen im Bild. Diese können zum Teil durch die **Distortion Coefficients** bei der Koordinatentransformation ausgeglichen werden.

$$DistortionCoefficients = [k_1, k_2, p_1, p_2, k_3, k_4, k_5, k_6]$$

- Die Koeffizienten **k_1 bis k_6** beschreiben die radialen Verzeichnungen im Bild.
- Die Koeffizienten **p_1** und **p_2** die tangentialen Verzeichnungen im Bild.

Rotation Matrix

Durch die Rotationsmatrix wird die unterschiedliche Ausrichtung des Kamerakoordinatensystems zum Weltkoordinatensystem beschrieben und umgekehrt.

$$RotationMatrix = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Translation Vector

Der Translationsvektor beschreibt die Verschiebung des Koordinatenursprungs vom Kamera- ins Weltkoordinatensystem und umgekehrt.

$$TranslationVector = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

Write Results

Die Ergebnisse werden in das Image Provider TcCOM-Objekt der Kamera geschrieben. Dadurch sind diese aus der SPS über die Methoden des Funktionsbausteins F_VN_GevCameraControl [▶ 1672] abrufbar.

Auch wenn noch keine Kalibrierung durchgeführt wurde, wird die Beschreibung des eingestellten Kalibriermusters geschrieben. Diese kann in der SPS mit der Methode GetCalibPatternRef [▶ 1680] abgerufen werden. Alternativ kann die Beschreibung des Kalibriermusters über die F_VN_GenerateCalibrationPatternReferencePoints [▶ 1680] auch direkt in der SPS erzeugt werden.

Save To File...

Die Ergebnisse können in einer json-Datei gespeichert werden. Alternativ können die Ergebnisse über den FB_VN_ReadCalibrationResult [▶ 1645] in die SPS eingelesen werden.

Undistort Selected

Auf das ausgewählte Bild werden die **Distortion Coefficients** angewendet und das Ergebnis in einem separaten Fenster angezeigt.

Show Output

Öffnet das Verzeichnis *C:\ProgramData\Beckhoff\Vision_CalibrationAssistantOutput*. In diesem werden Ergebnisbilder erstellt, um analysieren zu können, welche Merkmalspunkte gefunden und wie diese zugeordnet wurden.

Alternative in der SPS

Falls eine Kamera während der Maschinenlaufzeit kalibriert werden muss, steht dafür in der SPS die Funktion F_VN_CalibrateCamera [▶ 1102] zu Verfügung.

Kalibrierung in der SPS

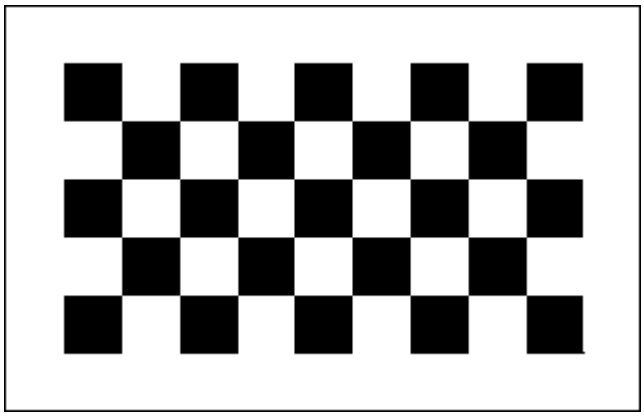
Die Dauer einer Kalibrierung kann wesentlich höher als ein SPS-Zyklus sein. Achten Sie daher darauf, dass in Ihrem Maschinenablauf keine Probleme verursacht werden, wenn Sie eine Kalibrierung in der SPS ausführen.

Beispiel zum Kalibrierungsassistent

Calibration Assistent [▶ 2925]

5.4.4.1 Schachbrettmuster

Ein Schachbrettmuster setzt sich aus abwechselnd hellen und dunklen Quadraten zusammen, welche zusammen ein rechteckiges Muster bilden. Zum Beispiel folgendes Schachbrett mit 4 x 8 Schnittpunkten:



Anforderungen an das Muster

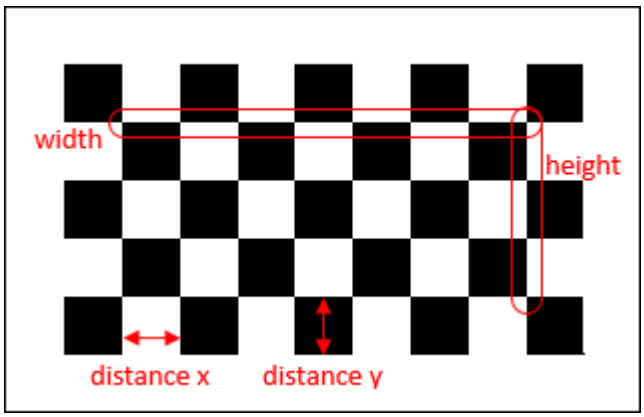
- Das Muster sollte auf einer möglichst ebenen (planaren) Fläche wie z.B. einer Glasscheibe aufgebracht sein.
- Die hellen und dunklen Quadrate müssen
  - in einem abwechselnden Muster angeordnet sein,
  - die gleiche Größe haben
  - und quadratisch sein.
- Ein Muster muss mind. 16 voll sichtbare Quadrate beinhalten, welche zusammen mind. 3 x 3 Schnittpunkte bilden.
- Es muss sich eine Ruhezone in Hintergrundfarbe um das Muster befinden.

Anforderungen an die Bildaufnahme

- Belichtung
  - Homogene Ausleuchtung des Musters
  - Guter Kontrast zwischen den hellen und dunklen Quadraten im Bild
  - Keine Öffnung im Schnittpunkt durch Überbelichtung
- Abbildung
  - Ein Quadrat muss durch mind. 15x15 Pixel abgebildet werden.

Konfiguration im Camera Calibration Assistant [▶ 116]

- **Width**
  Entspricht der Anzahl der Schnittpunkte in horizontaler Richtung.
- **Height**
  Entspricht der Anzahl der Schnittpunkte in vertikaler Richtung.
- **Distance x**
  Entspricht dem Abstand zweier Schnittpunkte in horizontaler Richtung.
- **Distance y**
  Entspricht dem Abstand zweier Schnittpunkte in vertikaler Richtung.



5.4.4.2 Symmetrische Kreismuster

Ein symmetrisches Kreismuster setzt sich aus Kreisen zusammen, welche gleichmäßig in Reihen und Spalten angeordnet sind.

Weitere unterstützte Muster mit Kreisen sind asymmetrische Kreismuster [▶ 128] und individuelle Kreismuster [▶ 129].

Anforderungen an das Muster

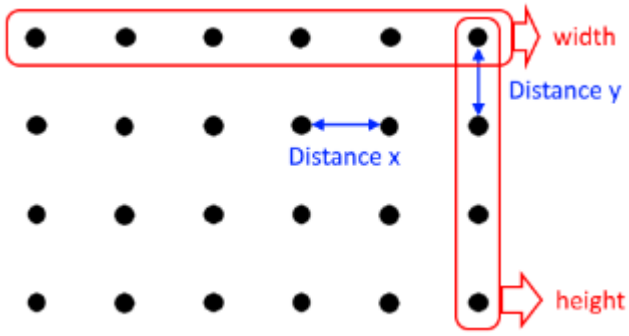
- Das Muster sollte auf einer möglichst ebenen (planaren) Fläche wie z.B. einer Glasscheibe aufgebracht sein.
- Die Kreise sollten möglichst gleich groß sein.
- Ein Muster muss aus mind. 9 Punkten bestehen.
- Die Kreismittelpunkte müssen horizontal und vertikal jeweils im gleichen Abstand aufgebracht sein. Der Abstand muss mind. so groß gewählt werden, dass die Kreise sich nicht berühren und als separate Objekte erkannt werden.
- Es muss sich eine Ruhezone in Hintergrundfarbe um das Muster befinden.

Anforderungen an die Bildaufnahme

- Belichtung
  - Homogene Ausleuchtung des Musters.
  - Guter Kontrast zwischen Kreisen und Hintergrund.
- Abbildung
  - Perspektivische Verzerrungen sollten geringgehalten werden.
  - Die Standard-Vorgaben der **Advanced Calibration Options** müssen eingehalten oder angepasst werden. So muss standardmäßig ein Kreis durch eine Fläche von mind. 25 Pixel und max. durch 15000 Pixel abgebildet werden. Die Fläche sollte nicht zu klein gewählt werden, da ansonsten auch Störungen vom Hintergrund erkannt werden.

Konfiguration im Camera Calibration Assistant [▶ 116]

- **Width**
  Entspricht der Anzahl Kreise pro Zeile.
- **Height**
  Entspricht der Anzahl Kreise pro Spalte.
- **Distance x**
  Entspricht dem Abstand zweier Kreismittelpunkte in horizontaler Richtung.
- **Distance y**
  Entspricht dem Abstand zweier Kreismittelpunkte in vertikaler Richtung.



- **Color inverted**
  Standardmäßig wird von dunklen Kreisen auf hellen Grund ausgegangen. Besteht das Muster stattdessen aus hellen Kreise auf dunklen Grund, so werden mit **Color inverted** die Farben getauscht.

5.4.4.3 Asymmetrische Kreismuster

Ein asymmetrisches Kreismuster ist ähnlich aufgebaut wie ein symmetrisches Kreismuster [▶ 127], nur dass jede zweite Zeile um eine halbe Spalte verrückt ist. Dadurch ist ein geringerer Abstand zwischen zwei Zeilen möglich, wodurch mehr Zeilen auf gleicher Fläche möglich sind.

Anforderungen ans Muster

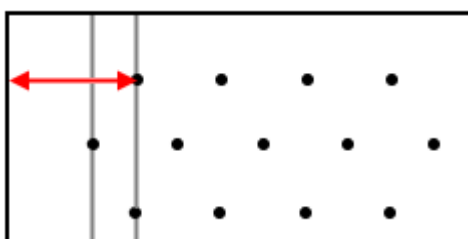
- Das Muster sollte auf einer möglichst ebenen (planaren) Fläche wie z.B. einer Glasscheibe aufgebracht sein.
- Die Kreise sollten möglichst gleich groß sein.
- Ein Muster muss aus mind. 9 Punkten bestehen.
- Die Kreismittelpunkte müssen horizontal und vertikal jeweils im gleichen Abstand aufgebracht sein. Der Abstand muss mind. so groß gewählt werden, dass die Kreise sich nicht berühren und als separate Objekte erkannt werden.
- Es muss sich eine Ruhezone in Hintergrundfarbe um das Muster befinden.

Anforderungen an die Bildaufnahme

- Belichtung
  - Homogene Ausleuchtung des Musters.
  - Guter Kontrast zwischen Kreisen und Hintergrund.
- Abbildung
  - Perspektivische Verzerrungen sollten geringgehalten werden.
  - Die Standard-Vorgaben der **Advanced Calibration Options** müssen eingehalten oder angepasst werden. So muss standardmäßig ein Kreis durch eine Fläche von mind. 25 Pixel und max. durch 15000 Pixel abgebildet werden. Die Fläche sollte nicht zu klein gewählt werden, da ansonsten auch Störungen vom Hintergrund erkannt werden.

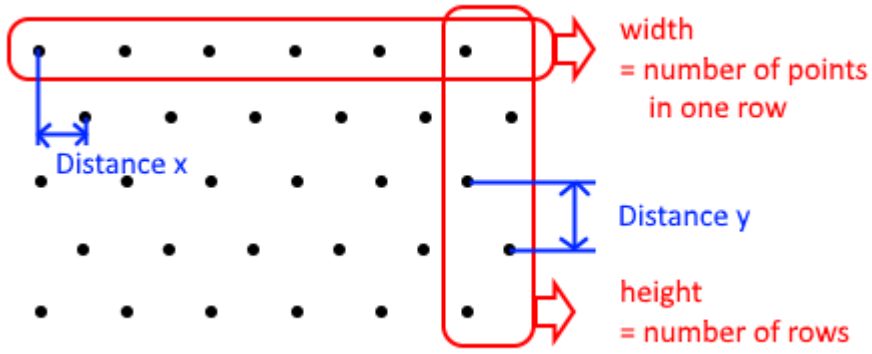
Konfiguration im Camera Calibration Assistant [▶ 116]

- **Start indent**
  Die erste Zeile startet später als die Zweite.



- **Width**
  Entspricht der Anzahl Kreise pro Zeile.

- **Height**
  Entspricht der Anzahl Kreise pro Spalte.
- **Distance x**
  Entspricht dem Abstand zum nächstgelegenen Kreismittelpunkt in horizontaler Richtung (zeilenübergreifend).
- **Distance y**
  Entspricht dem Abstand zum nächstgelegenen Kreismittelpunkt in vertikaler Richtung (innerhalb einer Spalte).



- **Color inverted**
  Wenn ein helles Muster auf dunklem Hintergrund gegeben ist.

5.4.4.4 Individuelle Kreismuster

Ein individuelles Kreismuster setzt sich aus Kreisen zusammen und bietet die Möglichkeit, die Position und damit die Anordnung der einzelnen Punkte des Musters selbst zu definieren. Dadurch ist es möglich, auch selbst gefertigte komplexe 3D-Muster für die Kalibrierung zu verwenden.

Anforderungen an das Muster

- Die Muster sollten auf möglichst ebenen (planaren) Flächen aufgebracht sein
- Die Kreise sollten möglichst gleich groß sein.
- Ein Muster muss aus mind. 9 Punkten bestehen.
- Der Abstand muss mind. so groß gewählt werden, dass die Kreise sich nicht berühren und als separate Objekte erkannt werden.
- Es muss sich eine Ruhezone in Hintergrundfarbe um das Muster befinden.

Anforderungen an die Bildaufnahme

- Belichtung
  - Homogene Ausleuchtung des Musters.
  - Guter Kontrast zwischen Kreisen und Hintergrund.
- Abbildung
  - Perspektivische Verzerrungen sollten geringgehalten werden.
  - Die Standard-Vorgaben der **Advanced Calibration Options** müssen eingehalten oder angepasst werden. So muss standardmäßig ein Kreis durch eine Fläche von mind. 25 Pixel und max. durch 15000 Pixel abgebildet werden. Die Fläche sollte nicht zu klein gewählt werden, da ansonsten auch Störungen vom Hintergrund erkannt werden.

Anforderung an die Beschreibungsdatei

- Die Beschreibung der Muster muss in einer XML-Datei erfolgen und auf dem Entwicklungsrechner in folgendem Ordner abgelegt werden: *C:\Users\Public\TcVision\CalibrationPattern*
- Die XML-Datei muss gegen das Schema TcVnCalibrationPattern.xsd im Ordner *%TwinCAT3DIR%\Components\Vision\TcVision* validiert werden.

- Nach den Vorgaben des Schemas muss die Beschreibungsdatei mindestens den DisplayName, den Origin und die 2D- oder 3D-Koordinaten der Kreispunkte enthalten.

Mittelpunkt

Da der Mittelpunkt automatisch durch die Beschreibung der Punkte gegeben ist, dient die Angabe des Origins nur als weitere Information, wird jedoch nicht in der Berechnung verwertet.

2D-Muster

Bei der Angabe von 2D-Mustern wird intern mit einer Z-Koordinate von 0 gerechnet.

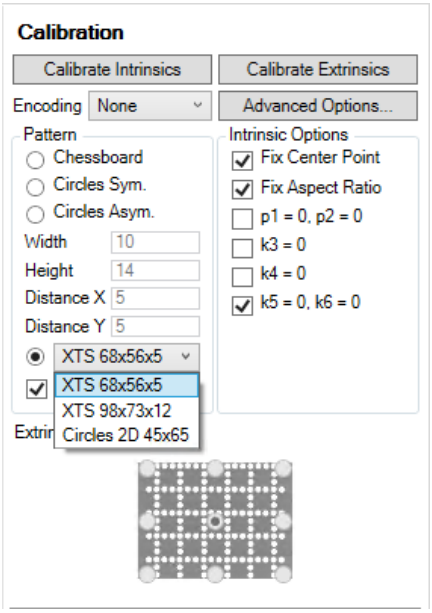
Konstantes Offset

Ein konstantes Offset kann als 3D-Muster mit konstantem Z-Wert realisiert werden.

Konfiguration im Camera Calibration Assistant [▶ 116]

Wenn gültige XML-Beschreibungsdateien in dem genannten Ordner gefunden wurden, werden diese mit dem Ausgelesenen DisplayName angezeigt.

- **Dropdown-Menü für individuelle Kreismuster**
  Auswahl eines individuellen Kreismusters.
- **Color inverted**
  Checkbox ob die Farbe des Bildes invertiert werden soll. Im Standard ohne Haken werden schwarze Objekte auf weißem Hintergrund erwartet.
- **Extrinsic Origin**
  Wird automatisch auf den Wert gesetzt, der in der Datei des ausgewählten Musters, hinterlegt ist und kann über die Oberfläche nicht verändert werden.
  - Das optionale Hintergrundbild für die schematische Darstellung des Kalibriermusters wird entsprechend angezeigt, sofern der Bildname inkl. Dateiendung in der Beschreibungsdatei angegeben wurde und die Bilddatei ebenfalls in dem gleichen Ordner liegt.



Beispielvorlage einer Beschreibungsdatei für individuelle Kreismuster

Der folgende Codeblock kann als Vorlage für die Erstellung eigener Beschreibungsdateien verwendet werden und enthält beispielhaft ausgefüllte Metadaten und 2D-Punktbeschreibungen. Die ebenfalls enthaltenen 3D-Punktbeschreibungen sind auskommentiert. Bei Verwendung müsste der Pattern2D Block vollständig durch den Pattern3D Block ersetzt werden, eine Vermischung ist nicht zulässig.

Zuordnung der Punkte

Es ist hilfreich, wenn die Punkte zeilen- oder spaltenweise sortiert sind und direkt zu der Lage der Punkte auf dem Kalibrierbild passen. Dadurch können die Punkte schneller und zuverlässiger zugeordnet werden.

Ausrichtung des Koordinatensystems

Das Koordinatensystem für die Kalibrierung muss dem Koordinatensystem der Bilder entsprechen und in X von links nach rechts und in Y von oben nach unten verlaufen. Das trifft auch auf die manuelle Kalibierung in der SPS und auf die Verwendung der F_VN_SortDetectedPatternPoints Funktion zu.

```
<?xml version="1.0" encoding="UTF-8"?>
<TcVnCalibrationPattern xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:noNamespaceSchemaLocation="file:///C:/TwinCAT/3.1/Components/Vision/TcVision/
TcVnCalibrationPattern.xsd">
    <MetaData>
        <DisplayName>12345</DisplayName>
        <Origin>Center</Origin>
        <PreviewImage>PreviewImageSample.png</PreviewImage>
        <Description>This is a calibration pattern</Description>
        <Revision>V1.0</Revision>
        <Width>10</Width>
        <Height>10</Height>
        <Depth>4</Depth>
        <NumberOfPoints>9</NumberOfPoints>
        <PointDiameter>1</PointDiameter>
    </MetaData>
    <Pattern2D>
        <!-- 1st row -->
        <Point2D x="-5" y="-5"/>
        <Point2D x="-0" y="-5"/>
        <Point2D x="5" y="-5"/>
        <!-- 2nd row -->
        <Point2D x="-5" y="0"/>
        <Point2D x="0" y="0"/>
        <Point2D x="5" y="0"/>
        <!-- 3rd row -->
        <Point2D x="-5" y="5"/>
        <Point2D x="0" y="5"/>
        <Point2D x="5" y="5"/>
    </Pattern2D>
    <!-- <Pattern3D> -->
        <!-- 1st row -->
        <!-- <Point3D x="-5" y="-5" z="-2"/> -->
        <!-- <Point3D x="-0" y="-5" z="-2"/> -->
        <!-- <Point3D x="5" y="-5" z="-2"/> -->
        <!-- 2nd row -->
        <!-- <Point3D x="-5" y="0" z="0"/> -->
        <!-- <Point3D x="0" y="0" z="0"/> -->
        <!-- <Point3D x="5" y="0" z="0"/> -->
        <!-- 3rd row -->
        <!-- <Point3D x="-5" y="5" z="2"/> -->
        <!-- <Point3D x="0" y="5" z="2"/> -->
        <!-- <Point3D x="5" y="5" z="2"/> -->
    <!-- </Pattern3D> -->
</TcVnCalibrationPattern>
```

5.4.5 GigE Vision Standard

TwinCAT Vision bzw. das Produkt TC3 GigE Vision Connector ist nach dem GigE Vision Standard (Version 2.0) zertifiziert. Somit kann TwinCAT Vision mit allen GigE Vision kompatiblen Kameras verwendet werden. Da der GigE Vision Standard bei jeder Version in Features unterteilt ist, die verpflichtend und optional sind, bedeutet eine Zertifizierung nicht, dass auch alle Features der Version unterstützt werden. Da Kameras ihrerseits auch zusätzliche Features haben dürfen, die nicht dem GigE Vision Standard entsprechen wie z. B. eine Datenkompression, kann auch aus der Richtung keine Aussage über die volle Unterstützung gemacht werden. Daher empfiehlt es sich, die benötigten Features im Einzelnen abzugleichen und zu testen.

Der GigE Vision Standard

GigE Vision ist ein Schnittstellen-Standard der industriellen Bildverarbeitung und ermöglicht die einfache Konfiguration und Bedienung von Industriekameras. Dabei steht GigE für Gigabit Ethernet. Durch die Nutzung des Gigabit Ethernet Kommunikationsprotokolls profitiert GigE Vision von folgenden Eigenschaften:

- Übertragungsrate ist standardmäßig 1 Gigabit/s (erweiterbar mit 2,5GigE, 5GigE und 10GigE).
- Mögliche Kabellängen bis zu 100 Meter ohne Verstärkung.
- Verwendung von standardmäßigen Hard- und Softwareschnittstellen für Gigabit Ethernet.

Der GigE Vision Standard definiert die Kommunikation zu kompatiblen Geräten per UDP/IP und besteht aus folgenden vier Elementen:

- Das GigE Vision Control Protocol (GVCP) definiert, wie GigE Vision Geräte anzusprechen sind. Es spezifiziert die Datenkanäle und Mechanismen für die Übertragung von Bildern und Konfigurationsdaten zwischen PC und Kamera.
- Das GigE Vision Stream Protocol (GVSP) spezifiziert die verschiedenen Datentypen und Übertragungsarten, um Bilder von einer Kamera zu einem PC zu transferieren. Zudem ist eine Packet Resend Option vorhanden, um Übertragungsfehler zu beheben.
- Der GigE Device Discovery Mechanism ermöglicht, Kameras per Suchanfrage im Netzwerk aufzufinden.
- Eine XML-Datei, genannt GenAPI-Beschreibung, die sämtliche öffentlichen Funktionen der Kamera definiert. Diese Beschreibung basiert auf dem GenICam-Standard.

Der GenICam Standard

Infos zum GenICam-Standard und zu der GenAPI-Beschreibung von Kameras finden Sie auf der Website der EVMA.

Kompatibilität

So finden Sie heraus, ob eine Kamera mit TwinCAT Vision kompatibel ist: Achten Sie darauf, ob in der Produktbeschreibung der Kamera explizit der Name „GigE Vision“ genannt oder das GigE Vision Logo gezeigt wird:

GiGE VISION

„GigE“ versus „GigE Vision“

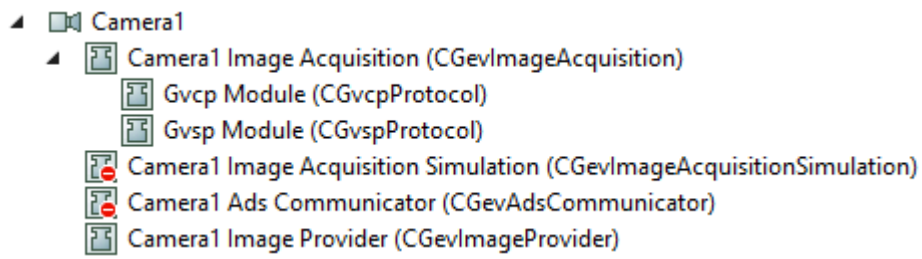
Achten Sie auf den Unterschied zwischen „GigE“ und „GigE Vision“! Nur Kameras mit GigE Vision Konformität werden unterstützt.

Verbindung zu anderen Geräten wird nicht unterstützt

Darüber hinaus lässt der GigE Vision Standard die Verbindung zu anderen Geräten (wie Blitzcontrollern) zu. Diese Funktion wird zurzeit von TwinCAT Vision nicht unterstützt.

5.4.6 TcCOM-Objekte

Jede GigE Vision Kamera-Instanz enthält im TwinCAT-Projekt eine Reihe von TcCOM-Objekten:



TcCOM-Objekte automatisch von TwinCAT Vision verwaltet

Grundsätzlich werden die TcCOM-Objekte automatisch von TwinCAT Vision verwaltet und müssen nicht manuell angepasst oder konfiguriert werden. Daher dienen folgende Informationen im Wesentlichen einem umfassenderen, technischen Verständnis.

Zyklische Tasks

Die TcCOM-Objekte `GevImageAcquisition`, `GevImageAcquisitionSimulation` und `GevAdsCommunicator` müssen mit einer zyklischen Task verknüpft sein, sofern sie aktiviert sind. Die entsprechende Konfiguration ist im Tab **Context** der Objekte zu finden. Im Normalfall müssen keine Tasks manuell verknüpft werden, da beim Anlegen eines Vision-Geräts benötigte Tasks automatisch erstellt und verknüpft werden. Details zum Erstellen und Verknüpfen von Tasks finden Sie im Kapitel CPU-Kerne und Tasks [▶ 63].

GigE Vision Image Acquisition

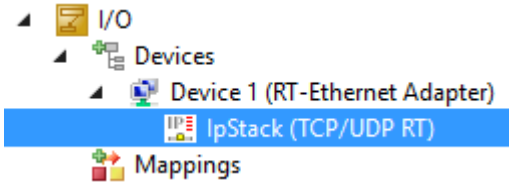
Das GevImageAcquisition-Objekt koordiniert die Kommunikation mit einer GigE Vision Kamera. Dazu kapselt und verwaltet es zwei Sub-Objekte, die die beiden Protokolle des GigE Vision [▶ 131] Standards implementieren:

- GigE Vision Control Protocol (GVCP)
- GigE Vision Streaming Protocol (GVSP)

Im Tab **Parameter (Online)** der beiden Sub-Objekte befinden sich Diagnoseinformationen zur Kamera-Kommunikation.

Zudem nutzt es einen UDP/IP-Stack auf einem entsprechenden Netzwerk-Adapter, um mit einer GigE Vision Kamera zu kommunizieren. In diesen Objekten können ebenfalls Diagnoseinformationen abgerufen werden. Details zu beiden Diagnosemöglichkeiten finden Sie im Anhang/Troubleshooting unter Kamerakommunikation [▶ 2964].

Die RT-Ethernet Adapter und IP-Stacks sind im TwinCAT-Projekt unter dem I/O Knoten zu finden:



Die Parametrierung der Image Acquisition sowie der beiden Protokolle (GVCP und GVSP) erfolgt im Configuration Assistant [▶ 79] im Konfigurationsbaum [▶ 83]. Eine Erklärung der Parameter finden Sie im Kapitel TcCOM-Parameter [▶ 106]. Dieses Objekt ist bei aktiver Kamera-Simulation deaktiviert.

GigE Vision Image Acquisition Simulation

Das GevImageAcquisitionSimulation-Objekt simuliert das Streaming-Verhalten einer GigE Vision Kamera und ist standardmäßig deaktiviert. Wenn der Simulationsmodus aktiv ist, ist stattdessen das GevImageAcquisition-Objekt deaktiviert.

ActivateBlockIDCorrection	Einstellung, ob statt originalen IDs künstlich generierte kontinuierliche Block-IDs verwendet werden sollen. Standard: deaktiviert

GigE Vision Ads Communicator

Das GevAdsCommunicator-Objekt wird beim Aufnehmen eines Kamera-Streams [▶ 109] benötigt und sendet entsprechend Bilder aus dem Router-Speicher per ADS an den TwinCAT Vision Service. Im Vergleich zum Abspeichern von Bildern über den Funktionsblock FB_VN_WriteImage wird für die Record-Funktionalität ein eigenes TcCOM-Objekt benötigt, da das Recording unabhängig von der SPS-Verarbeitung läuft.

Zudem wird im Config-Mode eine weitere Instanz des GevAdsCommunicator-Moduls benutzt, damit sämtliche Kamera-Assistenten mit der Kamera kommunizieren können.

PublishImages	Mit dieser Einstellung sendet der Ads Communicator jedes n-te Bild an den TwinCAT Vision Service. Diese Einstellung wird automatisch bei Aktivierung einer Stream-Aufzeichnung gesetzt und hängt vom Parameter **Image Offset** aus dem Record/Playback-Tab [▶ 109] ab. Er sollte nicht manuell gesetzt werden. Standard: 0

GigE Vision Image Provider

Das GevImageProvider-Objekt stellt die Schnittstelle zwischen einer Kamera und entsprechenden Funktionsblöcken in der SPS dar. Darunter fallen `FB_VN_GevCameraControl`, `FB_VN_SimpleCameraControl`, `FB_VN_ReadRegister` und `FB_VN_WriteRegister` sowie `FB_VN_ReadMemory` und `FB_VN_WriteMemory`. Dazu muss das Objekt im Karteireiter Symbol-Initialisierung der Projekt-Instanz mit den gewünschten SPS-Symbolen der Funktionsblöcke verknüpft werden. Sofern mehrere Kamera-Instanzen im Projekt existieren, muss anhand der Benennung darauf geachtet werden, dass das richtige GevImageProvider-Objekt ausgewählt wird.

ImageQueueSize	Anzahl an akquirierten Bildern, die zwischengespeichert werden sollen, wenn sie nicht direkt durch den Funktionsblock in der SPS entgegengenommen werden. Standard: 1

Verknüpfung der TcCOM-Objekte

Die folgende Übersichtsgrafik zeigt einen vereinfachten Zusammenhang der verschiedenen Objekte, die miteinander kommunizieren oder Daten abrufen.

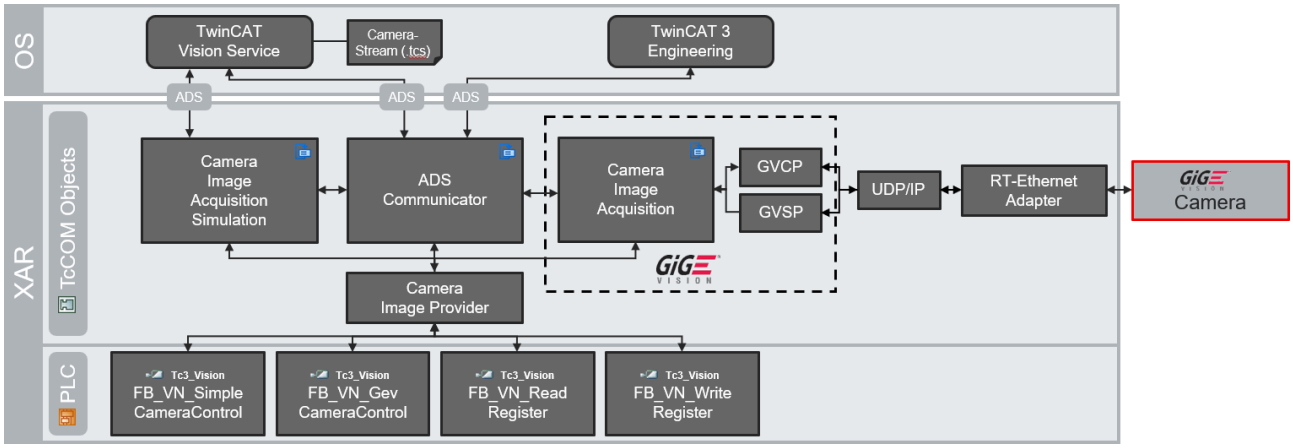


Abb. 5: Architektur der GigE Vision Camera Funktionalität

Die Verknüpfung der TcCOM-Objekte unterscheidet sich je nach Einstellung vom Simulationsmodus und dem Ads Communicator Objekt (siehe General Tab [▶ 74]). Die entsprechende Verknüpfungseinstellung ist im Karteireiter **Interface Pointer** der einzelnen Objekte zu finden. Standardmäßig wird die Verknüpfung automatisch basierend auf den Einstellungen in den Kamera-Assistenten vorgenommen und sollte nicht manuell verändert werden. Falls die Verknüpfungen aus Versehen verändert oder die TcCOM-Objekte erst im Nachhinein zur Kamera hinzugefügt wurden, können die Verknüpfungen durch einen Klick auf **Try to fix** im Karteireiter **General** repariert werden.

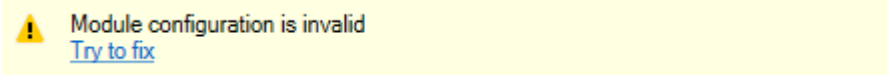
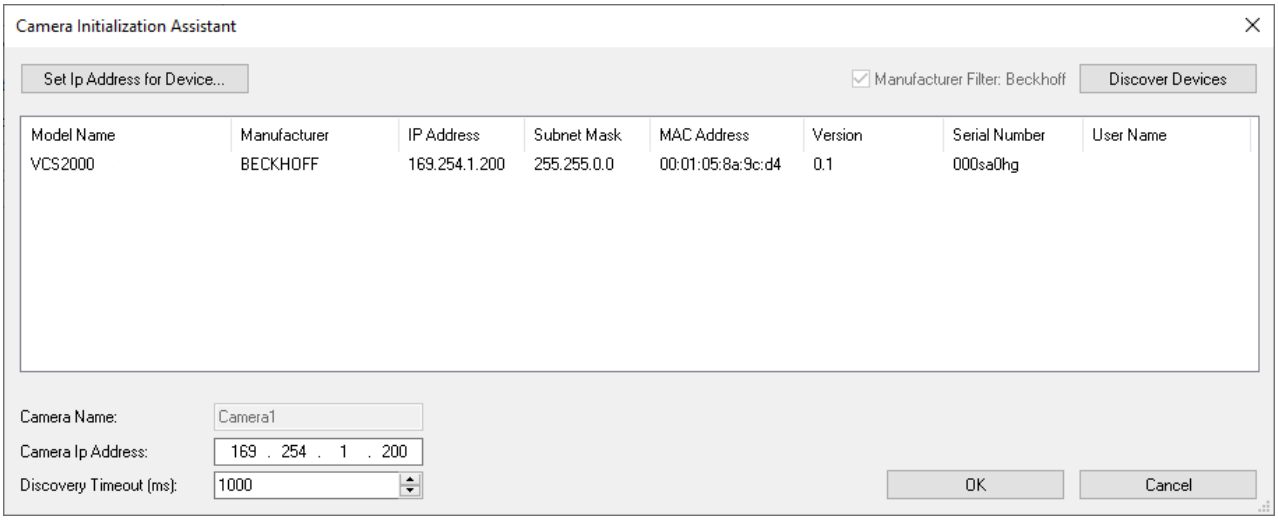


Abb. 6: Hinweis zu ungültiger Verknüpfung von TcCOM-Objekten

5.5 Beckhoff Camera-Objekte

Jedes Beckhoff Camera-Objekt repräsentiert jeweils eine Kamera innerhalb eines Projektes und ist durch eine andere Bezeichnung und Icon gekennzeichnet. Ein Projekt kann mehrere Kameraobjekte beinhalten. Da die Konfiguration und Bedienung identisch zum GigE Vision Camera-Objekt [▶ 72] erfolgt, können die Detailinformationen auf den entsprechenden Seiten dieses Kapitels eingesehen werden.

Mit einem Beckhoff-Camera-Objekt können nur Kameras vom Hersteller Beckhoff verknüpft und betrieben werden. Daher ist der Herstellerfilter im Camera Initialization Assistant immer aktiv und kann vom Benutzer nicht abgewählt werden. Beim Aufbau einer Kameraverbindung wird dies ebenfalls überprüft, so dass eine Verwendung im Configuration Assistent und im Run Mode nur mit Kameras vom Hersteller Beckhoff möglich ist.



5.6 File Source Objekte

Das File Source stellt eine alternative Bildquelle zur Kamera dar. Es ermöglicht, anstelle von Live-Bildern gespeicherte Bilder vom Dateisystem in die TwinCAT-Echtzeit zu laden. Beachten Sie, dass ein File Source im Vergleich zur Record/Playback [▶ 109] Funktion eines GigE Vision Kamera Objekts nicht zum Einladen von kontinuierlichen Kamera-Streams, sondern zum Einladen einzelner Bilder gedacht ist.

Dabei können Bilder verwendet werden, solange sie in einem Standard-Bildformat gespeichert sind, z.B. bmp, jpg, png.

Komprimierende Bildformate

Bei der Verwendung von komprimierenden Bildformaten hängt es davon ab, ob die angewendete Methode verlustbehaftet ist, da die Bilddaten dann nicht mehr den originalen Bilddaten der Kamera entsprechen. Daher empfehlen wir, BMP als Speicherformat zu nutzen.

Grauwertbilder können Farbbilder sein

Bilder, die wie Grauwertbilder aussehen, können mit mehreren Kanälen gespeichert sein. Achten Sie darauf beim Aufruf von Konvertierungsfunktionen.

Um Bilder in der SPS zu empfangen, existieren die Funktionsblöcke FB_VN_FileSourceControl [▶ 1664] und FB_VN_SimpleCameraControl [▶ 1697]. Über diese Funktionsblöcke erfolgt die Kommunikation zum File Source Control ähnlich der Kommunikation zu einer Kamera. Eine Alternative, um einzelne Bilder in die TwinCAT-Echtzeit zu laden, ist zudem der Funktionsblock FB_VN_ReadImage [▶ 1648].

Modi zur Bild-Bereitstellung

Das File Source Control hat zwei Modi zur Bereitstellung von Bildern:

1. Streaming Modus: Wie eine streamende Kamera werden neue Bilder in einem vorgegebenen zeitlichen Zyklus bereitgestellt.
2. Trigger Modus: Ein neues Bild wird bei einem Trigger-Signal bereitgestellt. Dieses kann entweder durch einen Software-Trigger oder durch manuelles Betätigen eines Buttons ausgelöst werden.

Die Einstellungen dazu sowie die Auswahl der Bilder erfolgen im File Source Control [▶ 136].

Anlegen eines File Source Knotens

Das Anlegen eines File Source Knotens ist in den First Steps [▶ 28] unter File Source Control anlegen [▶ 36] beschrieben.

Exportieren/Importieren eines File Source Knotens

Wenn Sie die Konfiguration eines File Source abspeichern oder in das Projekt einladen möchten, folgen Sie der Anleitung im Kapitel Exportieren/Importieren von Konfigurationen [▶ 144].

5.6.1 File Source Control

Die Konfiguration des File Source Control erfolgt in dem gleichnamigen Karteireiter:
VISION Node > File Source Node > Karteireiter File Source Control.

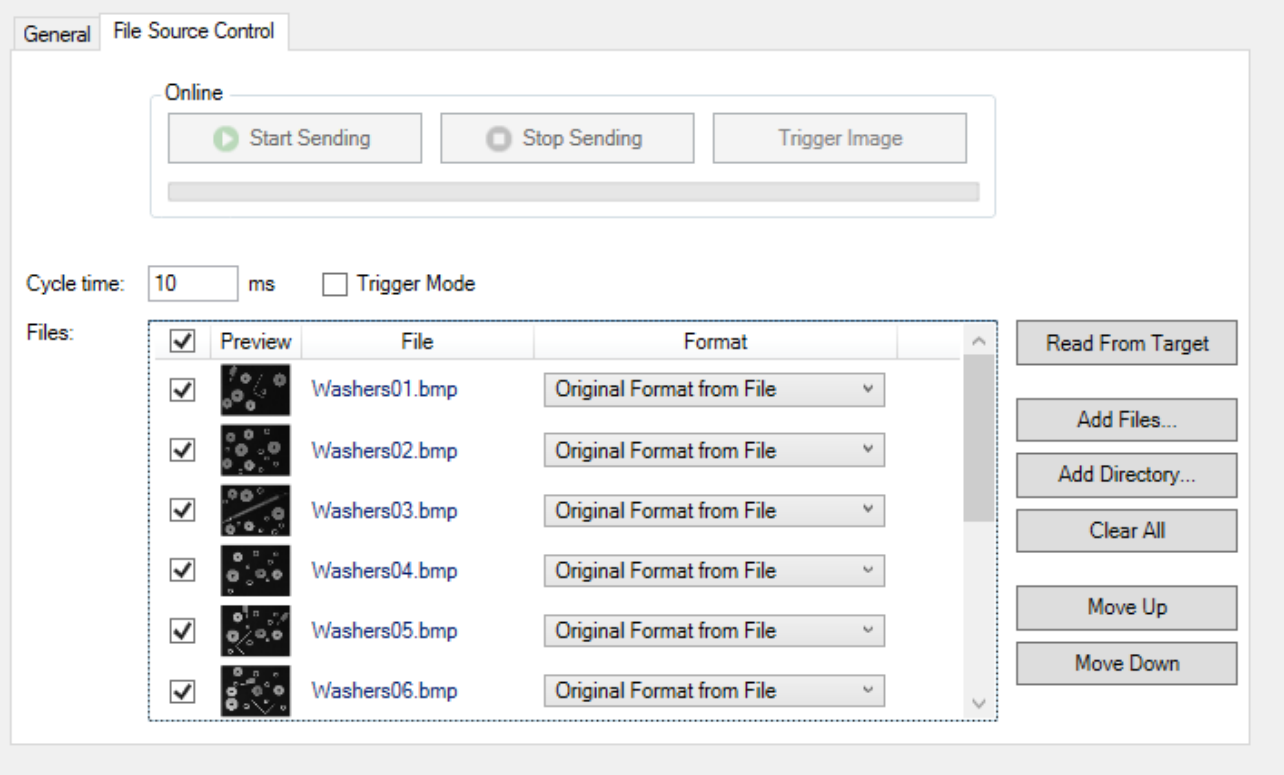


Abb. 7: File Source Control

Bilderauswahl verwalten

Die vom File Source Control zu verwendenden Bilder werden über eine Dateiliste und die dazu gehörigen Controls verwaltet:

Read from Target

Nach dem Erstellen eines File Source muss einmal **Read from Target** gedrückt werden und die Konfiguration aktiviert werden, damit Bilder hinzugefügt werden können.

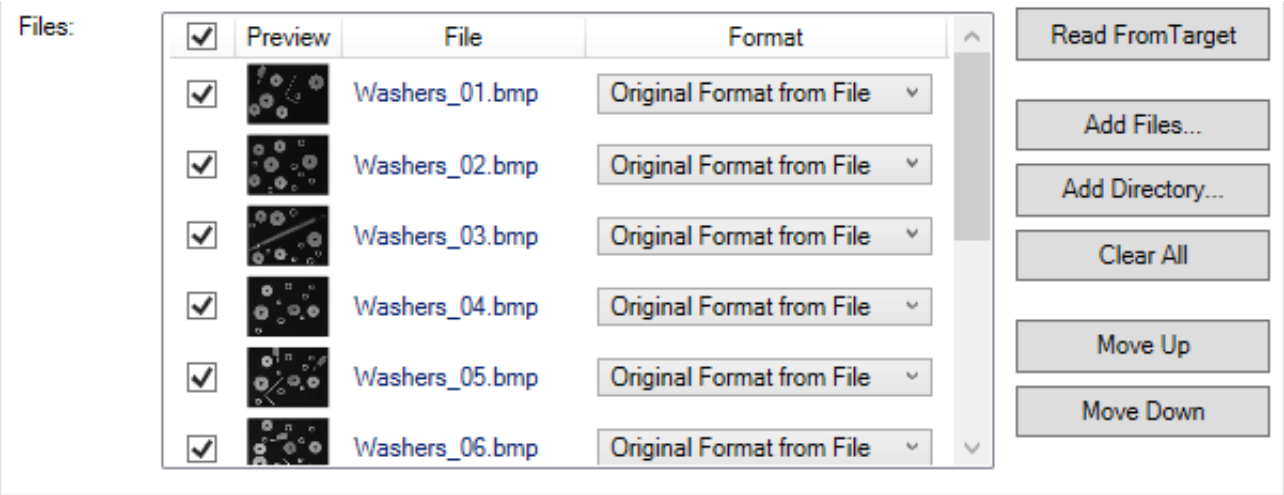


Abb. 8: Bildauswahl im File Source Control

Auswahl-Haken	Der Auswahlhaken in der ersten Spalte gibt an, ob ein Bild aktuell verwendet werden soll oder nicht.
Preview	Die Größe des Vorschaubildes ist abhängig von der Spaltenbreite.
File	Dateiname des Bildes inkl. Datentyp.
Format	Standardmäßig werden die Bilder im gespeicherten Format versendet. Alternativ kann vorgegeben werden, diese vorab in ein 1-Kanal Grauwertbild oder 3-Kanal RGB-Bild zu wandeln und dann zu versenden. **Wir empfehlen, dass Sie für jedes Bild explizit das Format angeben.**
Read From Target	Es wird ausgelesen, welche Bilder für das zugehörige File Source Control auf dem Zielsystem hinterlegt sind.
Add Files	Bilder vom Entwicklungssystem können über einen Datei Explorer ausgewählt werden. Im Hintergrund werden diese dann auf das Zielsystem kopiert, sodass diese dort unabhängig vom Entwicklungssystem zur Verfügung stehen. Bitte beachten Sie, dass nur Dateien mit eindeutigen Namen verwendet werden können. Zudem können an dieser Stelle nur Bildformate, keine Videoformate, eingefügt werden. Wenn Sie einen Kamera-Stream (*.tcs) in die SPS laden möchten, verwenden Sie bitte die Record/Playback [▶ 109] Funktion. **Falls nicht anders konfiguriert, werden die Bilddateien auf dem Zielsystem im Pfad „C:\Users\Public\TcVision\FileSources“ abgelegt. Stellen Sie sicher, dass der entsprechende Pfad nicht durch sonstige Software wie z.B. WriteFilter beeinträchtigt wird.**
Add Directory	Ein Verzeichnis des Entwicklungssystems, aus dem alle Bilder verwendet werden sollen, kann über einen Datei Explorer ausgewählt werden.
Clear All	Es werden alle Bilder aus der Dateiliste entfernt.
Move Up / Move Down	Ein ausgewähltes Bild wird in der Dateiliste eine Position nach oben / nach unten verschoben.

Kontextmenü

Über **STRG+Mausklick** können mehrere Bilder in der Dateiliste markiert werden. Über einen Rechtsklick auf eins der markierten Bilder können anschließend Einstellungen für alle markierten Bilder vorgenommen werden:

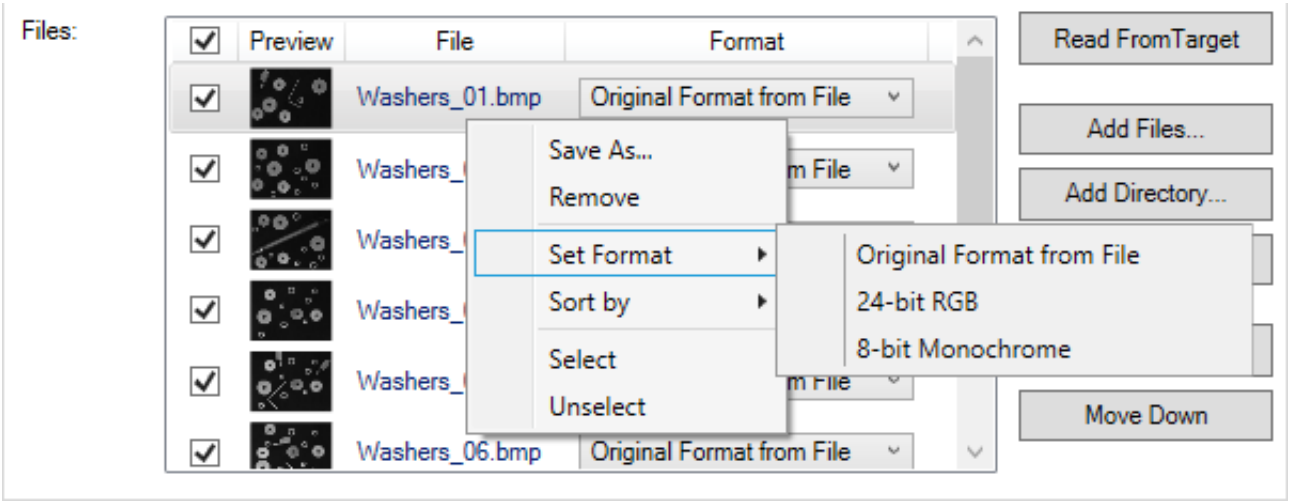


Abb. 9: Kontextmenü bei der Bildauswahl im File Source Control

Save As...	Öffnet einen Datei-Explorer-Dialog zum Speichern der markierten Bilder.
Remove	Löscht die markierten Bilder.
Set Format	Setzt das Bildformat für alle markierten Bilder: • **Original Format from File:** Das Format in dem das jeweilige Bild gespeichert ist, wird verwendet. • **24-bit RGB:** Die Bilder werden als 3-Kanal RGB Bild vom File Source versendet. • **8-bit Monochrome:** Die Bilder werden als 1-Kanal Grauwertbild vom File Source versendet.
Sort by	Sortiert alle Bilder der Dateiliste nach: • **Name (Ascending):** aufsteigend nach Name • **Name (Descending):** absteigend nach Name
Select	Wählt die markierten Bilder aus, sodass diese vom File Source Control gesendet werden können.
Unselect	Wählt die markierten Bilder ab, sodass diese vom File Source Control nicht gesendet werden.

Bild-Sende-Einstellungen

Das File Source Control kann wie eine streamende Kamera jeweils ein neues Bild in einem vorgegebenen zeitlichen Zyklus bereitstellen oder nach Aufruf eines Software-Triggers.

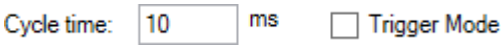


Abb. 10: Einstellungsmöglichkeiten im File Source Control

Cycle Time	Ist kein Trigger Mode ausgewählt, werden im Abstand der Cycle Time die Bilder an den Echtzeit-Kontext gesendet. Dabei wird die Reihenfolge der Bilder entsprechend der Dateiliste beachtet. Dieser Parameter ist vergleichbar mit der Framerate einer Kamera. **Die maximale Geschwindigkeit zum Einladen von Bildern hängt von dem System und von den Bildern ab. Die Cycle Time ist lediglich der gewünschte Wert.**
Trigger Mode	Bei ausgewähltem Trigger Mode wird nach Aufruf eines zugehörigen Software Triggers ein Bild gesendet. Beim Standard Software Trigger wird dabei die Reihenfolge der Bilder entsprechend der Dateiliste beachtet. Alternativ kann ein bestimmtes Bild anhand des Dateinamens getriggert werden.

Online

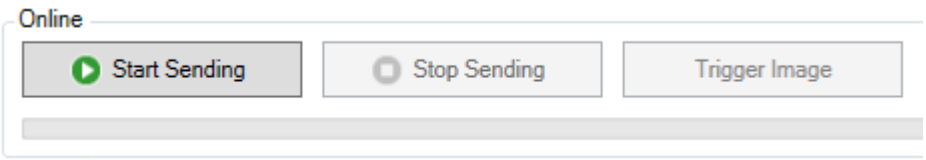
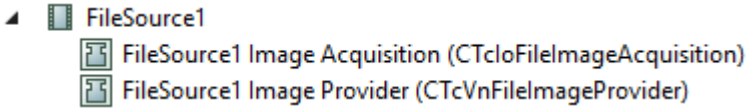


Abb. 11: Bedienung im File Source Control

Start Sending	**Start Sending** entspricht in der SPS der Methode StartAcquisition [▶ 1668] des Funktionsbausteins FB_VN_FileSourceControl [▶ 1664]. Ist kein Trigger ausgewählt, werden anschließend die Bilder im Abstand der **Cycle Time** vom File Source Control versendet. Ist der Trigger ausgewählt, so reagiert das File Source Control danach auf einen Software Trigger und versendet die Bilder entsprechend nach einem Trigger-Signal.
Stop Sending	**Stop Sending** entspricht in der SPS der Methode StopAcquisition [▶ 1669] des Funktionsbausteins FB_VN_FileSourceControl [▶ 1664] und beendet das Senden von Bildern durch das File Source Control.
Trigger Image	**Trigger Image** entspricht in der SPS der Methode TriggerImage [▶ 1671] des Funktionsblocks FB_VN_FileSourceControl [▶ 1664]. Hiermit können manuell Bilder getriggert werden. Dieser Button ist nur verfügbar, wenn der Trigger Mode aktiviert ist und das File Source am Senden ist.

5.6.2 TcCOM-Objekte

Jede FileSource Instanz enthält im TwinCAT-Projekt zwei TcCOM-Objekte:



TcCOM-Objekte werden automatisch von TwinCAT Vision verwaltet

Grundsätzlich werden die TcCOM-Objekte automatisch von TwinCAT Vision verwaltet und müssen nicht manuell angepasst oder konfiguriert werden. Daher dienen folgende Informationen im Wesentlichen einem umfassenderen, technischen Verständnis.

File Image Acquisition

Das TcIoFileImageAcquisition-Objekt koordiniert das Laden von Bildern aus dem Festplattenspeicher für FileSource-Instanzen. Dieses Objekt enthält folgende Parameter (zusätzlich zur `CycleTime` und `TriggerMode`, die im File Source Control [▶ 136] eingestellt werden können):

BaseDir	Dateipfad auf dem Zielsystem, in dem zur Auswahlliste hinzugefügte Bilder hinterlegt werden. Standard: *C:\Users\Public\TcVision\FileSources\<UniqueId>*
ServiceTimeout	Timeout zur Kommunikation mit dem TwinCAT Vision Service [▶ 68] in Millisekunden. Standard: 5000

Zyklische Tasks

Das File Image Acquisition Objekt muss mit einer zyklischen Task verknüpft sein. Die entsprechende Konfiguration ist im Karteireiter **Context** der Objekte zu finden. Im Normalfall müssen keine Tasks manuell verknüpft werden, da beim Anlegen eines Vision-Geräts benötigte Tasks automatisch erstellt und verknüpft werden. Details zum Erstellen und Verknüpfen von Tasks finden Sie im Kapitel CPU-Kerne und Tasks [▶ 63].

File Image Provider

Dieses Objekt stellt die Schnittstelle zwischen dem FileSource-Objekt und dem FileSource-Funktionsblock in der SPS dar. Die ObjectID-Variable des Funktionsblock FB_VN_FileSourceControl [▶ 1664] bzw. FB_VN_SimpleCameraControl [▶ 1697] muss nach dem Kompilieren des SPS-Codes im Tab **Symbol Initialization** der Projekt-Instanz mit dem entsprechenden File Image Provider TcCOM-Objekt verknüpft werden. Sofern mehrere FileSource-Instanzen im Projekt existieren, muss anhand der Benennung darauf geachtet werden, dass das richtige Objekt ausgewählt wird. Dieses Objekt enthält folgende Parameter:

ImageQueueSize	Anzahl an geladenen Bildern, die zwischengespeichert werden sollen, wenn sie nicht direkt durch den Funktionsblock in der SPS entgegengenommen werden. Standard: 1

Verknüpfung der TcCOM-Objekte

Für ein File Source Gerät muss lediglich der Image Provider mit der Image Acquisition verknüpft sein. Standardmäßig wird diese Verknüpfung automatisch hergestellt und sollte nicht manuell verändert werden. Die folgende Übersichtsgrafik zeigt einen vereinfachten Zusammenhang der verschiedenen Objekte, die miteinander kommunizieren oder Daten abrufen.

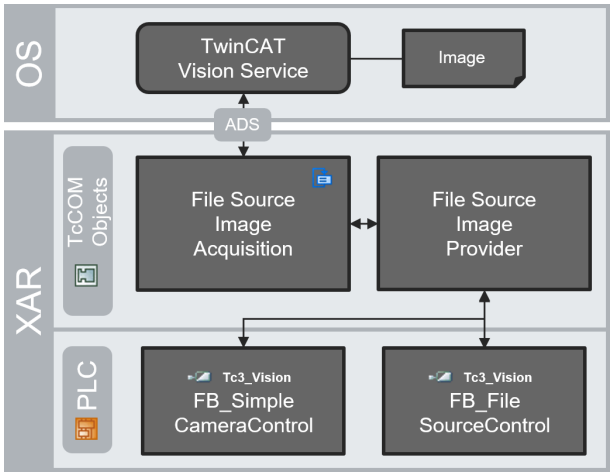
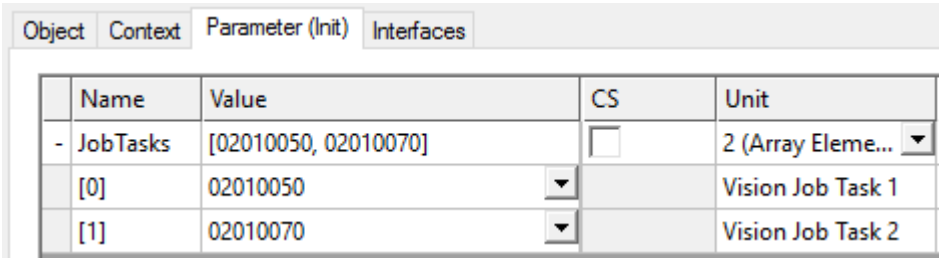


Abb. 12: Architektur der File Source-Funktionalität

5.7 Vision Job Pool

Der Vision Job Pool dient der Verwaltung von Job Tasks [▶ 66] für Bildverarbeitungsaufgaben. Dies ermöglicht die parallelisierte Ausführung von ausgewählten TwinCAT Vision Algorithmen, die in der API-Referenz [▶ 146] durch den folgenden Zusatz gekennzeichnet sind: “Can use available TwinCAT Job Tasks for executing parallel code regions”.

Durch einen Klick auf den **Vision Job Pool**, öffnet sich ein Fenster. Im Karteireiter **Parameter (Init)** erfolgt darin die Zuordnung der Job Tasks. In der ersten Zeile wird in der Spalte **Unit** angegeben wie viele Job Tasks zugewiesen werden sollen. Anschließend können den Array-Elementen unter **Job Tasks** über das Dropdown-Menü in der Spalte **Value** jeweils eine Job Task zugewiesen werden. Jede Job Task darf dabei nur einem einzigen Element zugeordnet werden.

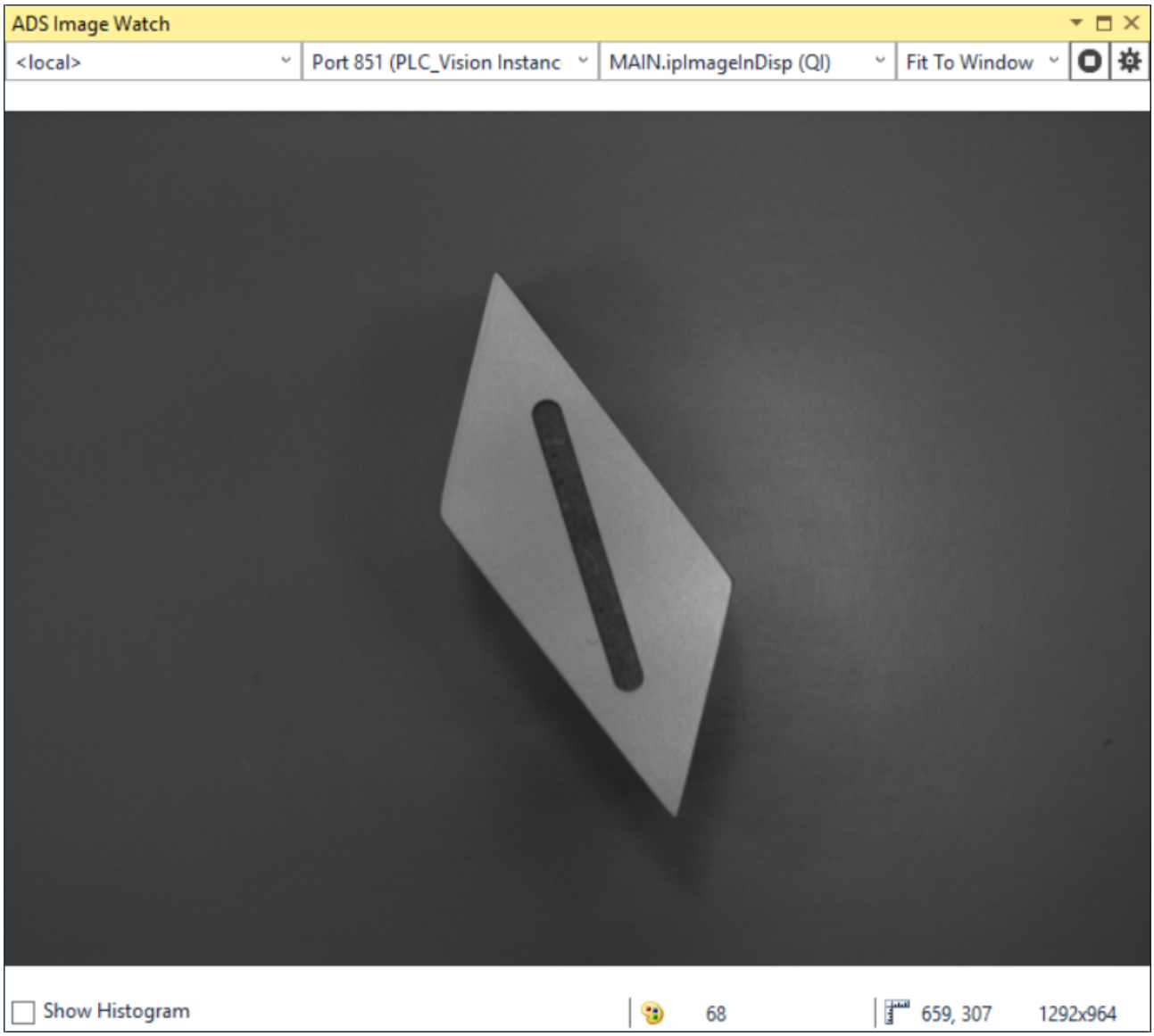


HINWEIS

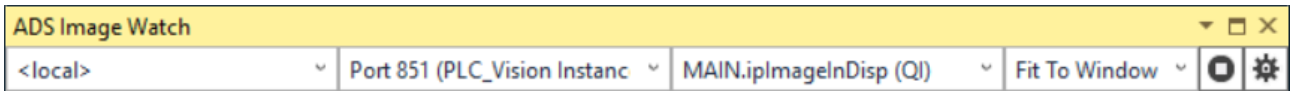
Beachten Sie bei der Erstellung von Job Tasks unbedingt die Einschränkungen und Empfehlungen [▶ 68]**.**

5.8 ADS Image Watch

Mit dem ADS Image Watch können Bilder vom Typ ITcVnDisplayableImage [▶ 426] aus der SPS abgerufen und in der Entwicklungsumgebung dargestellt werden. Öffnen Sie das ADS Image Watch über das Visual Studio Menü **TwinCAT > Windows > ADS Image Watch**.



Bildauswahl



Über die Auswahlfelder von links nach rechts erfolgt die Bildauswahl.

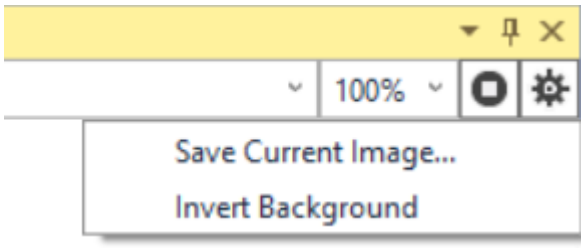
Zielsystem	Die Adressierung eines ADS Gerätes erfolgt über die ADS-AmsNetId und den ADS-Port. Die ADS-AmsNetId identifiziert dabei eindeutig das Zielsystem, welches als erstes ausgewählt werden muss. Standardmäßig ist das lokale System ausgewählt. Es können aber alle Systeme ausgewählt werden, zu denen eine Route angelegt ist, siehe Zielsystem auswählen [▶ 30].
Port	Die einzelnen Geräte auf dem System werden jeweils über eine ADS-Port Nummer identifiziert. Dabei ist für die SPS-Laufzeiten ein fester Port-Nummer-Bereich definiert, sodass die Auswahlliste bereits nach diesen gefiltert wird. Existiert nur ein SPS-Port auf dem Zielsystem, wenn dieses im ADS Image Watch ausgewählt wird, so erfolgt die Auswahl des SPS-Ports automatisch.
Bildvariable	Alle Variablen vom Typ ITcVnDisplayableImage [▶ 426] werden hier angezeigt und können ausgewählt werden. Unabhängig davon, ob die Variable zu dem Zeitpunkt auf ein Bild zeigt und dieses von einem unterstützten Bildformat [▶ 143] ist.
Größe	Ohne zusätzliche Einstellungen wird das Bild auf die Größe des aktuellen Fensters skaliert (**Fit To Window**). Es kann aber auch bewusst auf eine Größe in Relation zum Originalbild gesetzt werden. Dabei ist zu beachten, dass das Bild vor der Übertragung über ADS in der Echtzeitumgebung von der *PlcAuxTask* herunterskaliert wird, falls die ausgewählte Größe kleiner ist als das Originalbild. Wenn das Bild dann abgespeichert wird, hat es nicht die Originalgröße.
Stop / Play	Mit dem alternierenden Stop / Play Button lässt sich die Aktualisierung der Bildanzeige anhalten oder wieder starten. Wenn das Pause Icon zu sehen ist, ist die Aktualisierung aktiv und es werden kontinuierlich Bilder empfangen. Wenn das Play Icon zu sehen ist, wurde die Aktualisierung gestoppt und die Anzeige des letzten Bildes wird beibehalten. Es werden keine Bilder mehr empfangen. Dadurch ist es möglich, das Bild und jedes Detail z. B. durch Zoomen in Ruhe zu analysieren.

Bild Skalierung

Wenn die ausgewählte Größe ungleich 100% ist, wird das angezeigte Bild interpoliert. Bei **Fit To Window** oder einer Verkleinerung wird linear interpoliert und bei Vergrößerungen mit der Nearest-Neighbor-Interpolation.

Je nach Bildinhalt bzw. Pixelwerten kann es dabei zu Abweichungen kommen. Wenn einzelne Pixel im Detail analysiert werden sollen, sollte die Größe daher auf 100 % eingestellt werden.

Optionen



Über das **Action**-Icon, oben rechts neben der Bildgröße und dann **Save Current Image** kann das aktuell angezeigte Bild gespeichert werden.

Weiterhin kann der Hintergrund um das Anzeigebild mit **Invert Background** zwischen hell und dunkel gewechselt werden. Das kann bei sehr hellen oder dunklen Bildern hilfreich sein, damit der Bildinhalt oder Bildrand durch einen hohen Kontrast zum Hintergrund eindeutig zu sehen und zu unterscheiden ist.

Bildgröße

Das Bild wird in der Größe gespeichert, die das ADS Image Watch von der SPS abfragt. Um ein Bild in Originalgröße zu speichern, ist zuvor die Bildgröße auf mindestens 100% zu setzen oder das Fenster auf mindestens die originale Bildgröße zu ziehen. Bei einer eingestellten Bildgröße von größer als 100% wird das Bild ebenfalls in Originalgröße abgespeichert. Bei einer kleineren Einstellung oder bei **Fit to Window** und zu kleinem Fenster, wird das Bild in entsprechend reduzierter Größe gespeichert.

Unterstützte Bildformate

Im ADS Image Watch können folgende Bildformate in den folgenden Farbräumen angezeigt werden:

- **1-Kanal-Bilder** werden als **Grauwertbild** interpretiert. Hierbei werden alle Bilder mit Pixeltypen des Enums ETcVnElementType [▶ 210] unterstützt.
- **3-Kanal-Bilder** werden als **RGB-Farbbild** interpretiert. Hierbei werden Bilder mit 8 Bit oder 16 Bit pro Pixel pro Kanal unterstützt; das entspricht den Pixeltypen [▶ 210] USINT, SINT, UINT und INT.
- **4 Kanal Bilder** werden als **RGBA-Farbbild** interpretiert. Hierbei werden Bilder mit 8 Bit pro Pixel pro Kanal unterstützt; das entspricht den Pixeltypen [▶ 210] USINT und SINT.

Sonstige Bildformate

Andere Bildformate und Farbräume können in der SPS als ITcVnImage verarbeitet werden. Zur Darstellung müssen sie jedoch erst mittels F_VN_ConvertColorSpace in eins der genannten darstellbaren Formate konvertiert werden, bevor sie in ein ITcVnDisplayableImage umgewandelt werden. Erfolgt dies nicht, werden die Bilder entweder gar nicht oder in falscher Farbdarstellung angezeigt.

Darstellung von 16-Bit Bildern

16-Bit Bilder werden als 8-Bit Bildern von der SPS zum ADS Image Watch übertragen. Daher werden die Pixelwerte in der Statusleiste nur in 256er Inkrementen (0, 256, 512, etc.) angezeigt, beziehen sich dennoch auf 16 Bit: (0: Minimum, 65.280: Maximum).

Darstellung von (L)REAL Bildern

(L)REAL Bilder werden als 8-Bit Bilder von der SPS zum ADS Image Watch übertragen. Dafür wird der Wertebereich [-1, +1] auf [-127, +127] skaliert und als Integer übertragen. Fließkomma-Werte außerhalb dieser Grenzen werden als -1 bzw. +1 interpretiert. Im ADS Image Watch werden die übertragenden Pixelwerte wieder durch 127 geteilt, sodass Fließkommawerte im Bereich [-1, +1] angezeigt werden.

Statusleiste

Der Statusleiste unter dem Bild sind folgende Informationen zu entnehmen (von links nach rechts):

Abb. 13: Statusleiste im ADS Image Watch

- Farbwerte/Intensitätswerte an der Mauszeigerposition auf dem Bild
  oder Anzahl der Pixel im entsprechenden Behälter an der Mauszeigerposition im Histogramm
- Bildkoordinaten an der Mauszeigerposition auf dem Bild
  oder Behälter an der Mauszeigerposition im Histogramm
- Bildauflösung

Histogramm

Wenn das Kontrollkästchen **Show Histogramm** aktiviert ist, wird unten links im Bild das Histogramm des Bilds angezeigt. Dieses kann je nach Bildtyp mehrere überlagert dargestellte Kanäle haben.

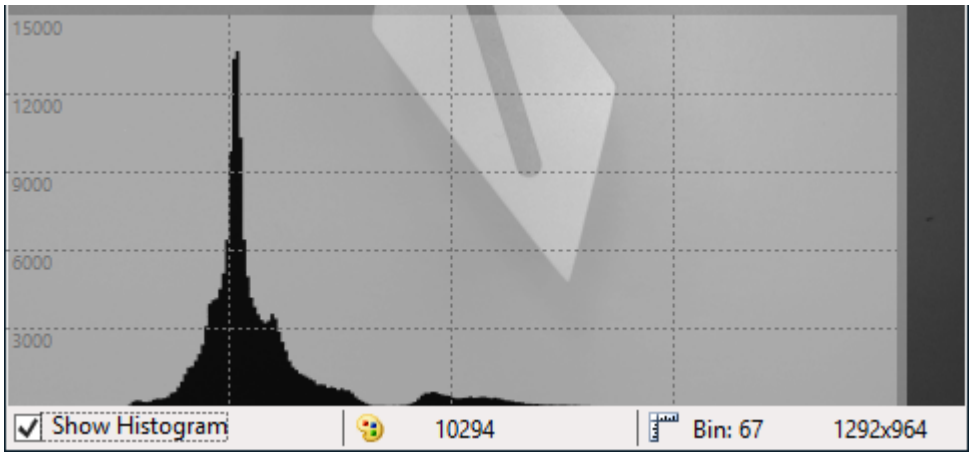


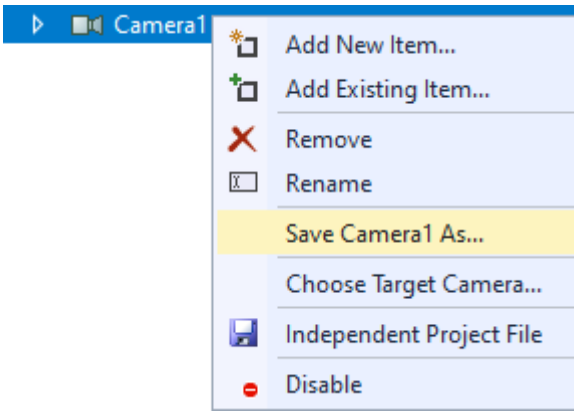
Abb. 14: Histogramm im ADS Image Watch

5.9 Exportieren/Importieren von Knoten

TwinCAT bietet die Möglichkeit, einzelne Konfigurationsknoten wie I/O- oder MOTION-Elemente in einer separaten XTI-Datei abzuspeichern bzw. aus dieser zu importieren. Dies ist ebenfalls mit Application-Knoten sowie mit GigE Vision Camera und File Source-Objekten möglich. Auf diese Weise können Konfigurationen für Vision-Geräte unabhängig vom Projekt gespeichert und an anderer Stelle wieder importiert werden.

Exportieren

1. Tätigen Sie einen Rechtsklick auf den zu exportierenden Knoten und klicken Sie auf **Save <Name> As…**

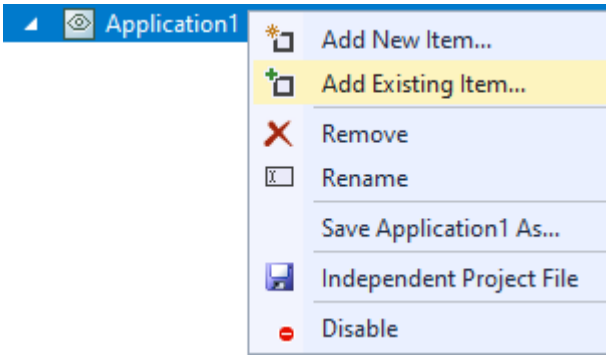


2. Wählen Sie einen Speicherort für die XTI-Datei aus.

Importieren

Je nachdem, ob Sie einen ganzen Application-Knoten oder nur ein einzelnes Vision-Gerät importieren möchten, führen Sie folgende Schritte auf dem VISION Knoten oder auf einem entsprechenden Application-Knoten aus.

3. Tätigen Sie einen Rechtsklick auf den Knoten, in den Sie die Konfiguration importieren möchten und klicken Sie auf **Add Existing Item…**



4. Wählen Sie die zu importierende XTI-Datei aus.
5. Tätigen Sie ggfs. System-abhängige Einstellungen (z. B. Netzwerkadapter) für die importierten Vision-Geräte.

Duplizieren von Kamera-Konfigurationen

Bitte beachten Sie, dass eine erfolgreiche Übernahme der gleichen Konfiguration für mehrere Kameras nur dann sichergestellt werden kann, wenn sowohl das Kameramodell als auch die Firmware exakt identisch sind. Anderenfalls können bspw. veränderte Register-Adressen der Kamera zu Problemen führen.

Unabhängige Projektdateien

Es ist ebenfalls möglich, die Konfigurationsknoten nicht gänzlich zu exportieren, sondern innerhalb des Projekts in einer eigenen Datei zu speichern. Dadurch wird die Konfiguration der entsprechenden Knoten nicht in der TwinCAT Projektdatei gespeichert, sondern in einer eigenen XTI-Datei. Dies hilft beispielsweise bei Versionskontrollsystemen, sodass bei der Modifikation einer Konfiguration nicht die gesamte Projektdatei verändert wird.

6 API-Referenz

Dieses Kapitel beschreibt die TwinCAT Vision API, d.h. sämtliche Möglichkeiten zur programmatischen Nutzung der TwinCAT 3 Vision Bibliothek. Da TwinCAT Vision in der TwinCAT 3 SPS und in TwinCAT 3 C++ verwendet werden kann, stellen diese beiden Dokumentationen die Grundlage dar. Alle TwinCAT Vision spezifischen Beschreibungen sind daher in die beiden folgenden Unterkapitel aufgeteilt.

6.1 SPS

Diese Kapitel beinhaltet die TwinCAT Vision API-Beschreibung zur programmatischen Verwendung der Bibliothek in der SPS. Da TwinCAT Vision einzelne Neuerungen mit sich bringt, ist das Softwarekonzept explizit erklärt:

- Softwarekonzept [▶ 146]
- Datentypen [▶ 170]
  (Aliase, Konstanten, Arrays, Enums, Strukturen)
- Interfaces [▶ 259]
  (Bilder, Container, etc.)
- Funktionen [▶ 448]
  (Bildverarbeitung)
- Funktionsblöcke [▶ 1620]
  (Kommunikation mit Kameras und dem Dateisystem, Bildverarbeitung)

6.1.1 Softwarekonzept

Dieses Kapitel beschreibt die Neuerungen bzw. die Besonderheiten, die TwinCAT Vision mit sich bringt. Die Grundlagen der SPS- Programmierung in TwinCAT 3 sowie die Referenz Programmierung finden Sie in der allgemeinen TwinCAT 3 Dokumentation. Innerhalb der Beschreibung zur PLC finden Sie ein Kapitel mit TwinCAT-3-Programmierkonventionen. Die darin enthaltenen Empfehlungen, Hinweise sowie Vor- und Nachteile ermöglichen es, einen einheitlichen Programmaufbau, eine konsistente Benennung von Objekten, Variablen, Instanzen und einen leichter lesbaren und verständlicheren Code zu erreichen und damit die Entwicklung, Nutzung und Wartung der Programme zu vereinfachen.

Eindeutiger Zugriff auf Bibliotheksmodule

Wenn die Tc3_Vision Bibliothek in eine andere Bibliothek eingebunden (referenziert) werden soll und somit mehrfach in einem Projekt vorkommen kann, muss ein eindeutiger Zugriff über den Namensraum als Präfix des Bezeichners erfolgen. Im Standard z. B. `hr := Tc3_Vision.F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, hr);`

Für weitere Informationen und Einstellmöglichkeiten schauen Sie die SPS-Projekt Einstellungen und die Bibliothekseigenschaften an.

⚠ WARNUNG

Zugriff auf dieselben Daten bei der Verwendung von mehreren Tasks

Wenn von mehreren Tasks z. B. auf dasselbe Bild zugegriffen werden soll, müssen diese Zugriffe unbedingt nach diesen Richtlinien synchronisiert werden. Ansonsten besteht die Gefahr eines inkonsistenten oder ungültigen Datensatzes was zu unabsehbaren Folgen und Fehlverhalten des Programms führen kann.

6.1.1.1 Interface Pointer

Für TwinCAT Vision werden dynamische Speicherbereiche benötigt, um z. B. Bilder und Container zu speichern. Diese Bilder und Container werden durch Variablen vom Typ ITcVnImage [▶ 426] und ITcVnContainer [▶ 385] repräsentiert. Da es sich dabei um Interfaces handelt und diese auf einen dynamischen Speicherbereich zeigen, werden diese Variablen als Interface Pointer bezeichnet.

Bei den TwinCAT Vision API-Funktionen wird die zugehörige Speicherverwaltung innerhalb der Funktionen automatisch geregelt, hier ist nichts zu beachten.

Beispielhaft wird ein neues Bild angelegt (800x600 Pixel vom Typ `TCVN_ET_USINT` mit einem Kanal):

```
VAR
    hr        : HRESULT;
    ipImage   : ITcVnImage;
END_VAR

hr := F_VN_CreateImage(ipImage, 800, 600, TCVN_ET_USINT, 1, hr);
```

Dieses Bild soll nun in ein RGB-Bild mit 3 Kanälen umgewandelt werden. Daher wird auch 3-mal so viel Speicher benötigt. Dafür genügt der folgende Funktionsaufruf:

```
hr := F_VN_ConvertColorSpace(ipImage, ipImage, TCVN_CST_GRAY_TO_RGB, hr);
```

Intern gibt er den bisherigen Speicherbereich frei und allokiert neuen Speicher in passender Größe. Interface Pointer sind also eine einfache Möglichkeit, Objekte mit dynamisch benötigtem Speicher in der SPS zu verwalten.

⚠ WARNUNG

Konsequenzen von Fehlgebrauch

Wenn Sie folgende Hinweise zum Gebrauch von Interface Pointern nicht beachten, kann dies zu **vollständigen Systemabstürzen** und zu **Speicher-Lecks** führen!

Referenzzähler

Sobald die Variablen nicht mehr ausschließlich als Funktionsparameter für die TwinCAT Vision API-Funktionen verwendet werden, ist es jedoch notwendig, sich genauer mit dem Konzept von Interface Pointern vertraut zu machen.

Es ist wichtig zu wissen, dass die Variable vom Typ Interface Pointer nur ein Zeiger auf ein Objekt im Speicher ist. Weist man also eine Variable einer anderen zu, so wird lediglich der Pointer auf diesen Speicherbereich kopiert, nicht jedoch die Daten selbst.

Das bedeutet, wenn die Daten in einer der beiden Variablen verändert werden, sind sie auch automatisch in der anderen geändert.
Wenn der Speicher in einer der Variablen freigegeben wird (z. B. im obigen Beispiel F_VN_ConvertColorSpace [▶ 1270]) und die zweite Variable immer noch ihren Pointer auf diesen Speicherbereich hat, so führt der Zugriff auf diesen bereits freigegebenen Speicher oftmals zu einem kompletten Systemabsturz.

Aus diesem Grund haben die Objekte im Speicher alle einen Referenz-Zähler. Sobald eine Variable einer anderen zugewiesen wird, muss dieser Referenz-Zähler erhöht werden. Wird einer Variablen ein neuer Pointer zugewiesen, muss der Referenz-Zähler auch wieder verringert werden.

Sobald der Referenz-Zähler dabei den Wert 0 erreicht, wird der Speicher freigegeben. Die entsprechenden Daten sind somit vollständig gelöscht.

Um anzuzeigen, ob eine Pointer-Variable auf einen gültigen Speicherbereich zeigt, muss dieser Pointer immer auf 0 gesetzt werden, wenn er nicht mehr verwendet wird (und daher der Referenz-Zähler verringert wurde). Standardmäßig werden alle angelegten Variablen (und damit auch die Pointer) in der PLC initial beim Programmstart automatisch auf 0 gesetzt.

Verwaltung von Interface Pointern

In den TwinCAT Vision API-Funktionen (gekennzeichnet durch F_VN_...) wird der Referenzzähler von Interface Pointern automatisch verwaltet. Bei ihrer Verwendung ist also auf nichts Spezielles zu achten. Wenn Interface Pointer jedoch manuell kopiert werden oder z. B. als Funktionsparameter verwendet werden, ist eine manuelle Verwaltung nötig (siehe unten).

Validität von Interface Pointern prüfen

Da der Wert 0 bei einem Interface Pointer prinzipiell auch nur einem ungültigen Speicherbereich entspricht - und ein Zugriff darauf ebenfalls zu einem kritischen Systemfehler führt - muss vor jeder Verwendung geprüft werden, ob der Wert ungleich 0 ist.

In den TwinCAT Vision API-Funktionen (`F_VN_...`) geschieht diese Prüfung und entsprechende Behandlung intern automatisch. Bei Interface-Methoden, wie z. B. `TcRelease()` im folgenden Beispiel, ist dies jedoch nicht möglich, da bereits deren Aufruf zu einem Fehler führt, wenn die Variable 0 ist. Daher muss die Variable dort vorher explizit geprüft werden.

⚠ WARNUNG

Methodenzugriff auf ungültigen Speicherbereich führt zum kompletten Systemabsturz

Bei Interface-Methoden ist eine interne Überprüfung nicht möglich, da die zugehörigen Objekte existieren müssen, um die Methode aufrufen zu können.

Stellen Sie selbst außerhalb des Methodenaufrufs sicher, dass das entsprechende Objekt vorhanden ist (Interface Pointer ungleich 0).

Zudem tritt ein Systemabsturz nicht zwangsweise immer auf. Falls der ungültige Speicherbereich zufällig ungenutzt bleibt, kann ein Systemabsturz ausbleiben.

Freigeben eines Interface Pointers

Das Freigeben einer Interface Pointer Variablen besteht aus der Prüfung, ob diese bereits 0 ist und ansonsten Verringern des Referenz-Zählers und setzen der Variable auf den Wert 0. Da dies ein häufiger Anwendungsfall ist, gibt es die Funktion `FW_SafeRelease`, die alle benötigten Schritte mit einem einzigen Aufruf durchführt:

```
hr := FW_SafeRelease(ADR(ipImage));
```

Die interne Funktionsweise der `FW_SafeRelease` Funktion wird durch folgenden Code-Ausschnitt verdeutlicht:

```
IF ipImage <> 0 THEN        // Check if interface pointer is already = 0
                            // If not, call .TcRelease() and set it to 0
    ipImage.TcRelease();    // decrement reference count
    ipImage := 0;           // set to 0
END_IF
```

Kopieren eines Interface Pointers

Beim Kopieren eines Interface Pointers muss dessen Referenzzähler manuell erhöht werden. Als Kopiervorgang zählt sowohl eine direkte Zuweisung mittels des `:=` Operators, als auch ein Programm- oder Funktionsblock-Aufruf, bei dem ein Interface Pointer als Input- oder Output-Wert übergeben wird. Ebenfalls zählt eine Übergabe mittels I/O-Abbild. Zum Erhöhen des Referenzzählers steht folgende Methode zur Verfügung:

```
ipImage.TcAddRef(); // increment reference count
```

Zudem muss darauf geachtet werden, dass der zu beschreibende Interface Pointer vor dem Kopieren freigegeben ist. Anderenfalls entstehen Speicher-Lecks. Ein manueller Kopiervorgang eines Interface Pointers sieht folglich z. B. so aus:

```
IF ipSrcImage <> 0 THEN                 // check if source pointer is valid
    FW_SafeRelease(ADR(ipDestImage));   // release destination pointer
    ipDestImage := ipSrcImage;          // assign source pointer to destination pointer
    ipSrcImage.TcAddRef();              // increment reference count
END_IF
```

Beachten Sie, dass dabei nur der Interface Pointer, jedoch nicht das dahinterliegende Bild kopiert wird.

Faustregeln

Halten Sie sich bei der Handhabung von Interface Pointern an folgende Faustregeln.

Dabei ist egal, ob es sich bei den Interface Pointern um (anzeigbare) Bilder, Container, Iteratoren, etc. handelt, sie sind in den Beispielen gegenseitig auswechselbar.

Auf einem Interface Pointer eine Methode aufrufen

Es muss sichergestellt sein, dass der Interface Pointer valide ist
(Prüfung auf `<> 0`):

```
IF ipImage <> 0 THEN
    ipImage.GetImageInfo(stImageInfo);
END_IF
```

Grundsätzlich wird empfohlen, statt einer Interface-Methode die alternative `F_VN_`-Funktion anzuwenden (falls diese existiert), denn dann wird die Prüfung automatisch intern ausgeführt:

```
hr := F_VN_GetImageInfo(ipImage, stImageInfo, hr);
```

Wenn Sie die Prüfung und den Aufruf der Interface-Methode im selben Ausdruck durchführen, müssen Sie `AND_THEN` statt `AND` verwenden. Nur in dem Fall wird der hintere Teil nur ausgeführt, wenn der vordere Teil `TRUE` liefert:

```
IF ipIterator <> 0 AND_THEN ipIterator.CheckIfEnd() <> S_OK THEN
    ipIterator.GetContainer(ADR(ipElement));
END_IF
```

Interface Pointer manuell kopieren

Stellen Sie zuerst sicher, dass der zu beschreibende Interface Pointer freigegeben ist. Erhöhen Sie nach dem Kopieren den Referenzzähler des Interface Pointers:

```
FW_SafeRelease(ADR(ipDestImage));
IF ipSrcImage <> 0 THEN
    ipDestImage := ipSrcImage;
    ipSrcImage.TcAddRef();
END_IF
```

Interface Pointer im VAR_INPUT deklarieren

Bei Funktionen und Methoden muss nichts beachtet werden. Bei Programmen und Funktionsblöcken bleibt der Pointer in den Variablen auch nach dem Aufruf erhalten und sollte daher am Ende der POU auf 0 gesetzt werden:

```
VAR_INPUT
    ipSrcImage: ITcVnImage;
END_VAR
//… functional code
ipSrcImage := 0;
```

Damit wird sichergestellt, dass beim nächsten Aufruf nicht mehr auf den Speicherbereich zugegriffen werden kann, da dieser außerhalb bereits freigegeben worden sein könnte. Eine Erhöhung des Referenzzählers innerhalb der POU ist keine Lösung, da bei einer neuen Zuweisung der vorige Pointer überschrieben wird und daher nicht mehr freigegeben werden kann, wodurch ein Speicherleck entsteht. Falls der Interface Pointer behalten werden soll, sollte er in eine lokale Variable kopiert werden.

Interface Pointer im VAR deklarieren

Geben Sie den Interface Pointer am Ende von Funktionen und Methoden frei. Der Interface Pointer existiert nach Beendigung der POU nicht mehr, daher muss vorher der Referenzzähler verringert werden. Bei Programmen und Funktionsblöcken sollte der Interface Pointer freigegeben werden, sobald er nicht mehr benötigt wird:

```
VAR
    ipImageWork: ITcVnImage;
END_VAR
//… functional code
hr := FW_SafeRelease(ADR(ipImageWork));
```

Nicht empfohlen - Interface Pointer im VAR_OUTPUT deklarieren

Die Deklaration eines Interface Pointers im VAR_OUTPUT ist nicht empfohlen, da die Verwendung von Outputs optional ist und so Speicherlecks entstehen können.
Bei Programmen und Funktionsblöcken bleibt der Pointer in den Variablen auch nach dem Aufruf erhalten, so dass bei Änderungen außerhalb, im Baustein auf einen freigegebenen Speicherbereich zugegriffen werden kann.
Damit das nicht passiert, müssen Sie immer sicherstellen, dass die Outputs verwendet werden und bei Programmen und Funktionsblöcken nach dem Kopieren der Output-Variable der Referenzzähler erhöht oder die Output-Variable auf 0 gesetzt wird.
Das ist sehr umständlich und fehlerträchtig, da man selbst darauf achten muss, an jeder Stelle im Code die Implementierung richtig umzusetzen. Daher wird stattdessen die Übergabe als Referenz im VAR_INPUT empfohlen.

HINWEIS

Speicherlecks

Die Verwendung eines Interface Pointers im VAR_OUTPUT ist nicht empfohlen. Bei Nichtbeachtung der vorstehend genannten Punkte können Speicherlecks und Systemabstürze entstehen.

Interface Pointer als Referenz im VAR_INPUT deklarieren

Die Übergabe als Referenz sollte verwendet werden, wenn eine Änderung des Interface Pointers (z. B. neues Bild / Container zuweisen) innerhalb der POU nach außen gereicht werden soll, um dort weiterverwendet werden zu können.

```
VAR_INPUT
    ipDestImage: REFERENCE TO ITcVnImage;
END_VAR
```

Das Beispiel Selbstgeschriebene Funktionen [▶ 2917] erläutert eine korrekte Verwendung von Interface Pointern in selbstgeschriebenen Funktionen und Methoden.

Konsequenzen von Fehlgebrauch

Wenn eine Interface Pointer Variable kopiert wird, dabei jedoch der Referenzzähler nicht mit `TcAddRef()` angepasst wird, gibt es zwei Pointer; dem Programm ist jedoch nur einer bekannt. Daher werden die Daten, auf die beide Interface Pointer Variablen zeigen, schon dann gelöscht, wenn nur eine der beiden Variablen freigegeben wird. Dabei kann das Freigeben sowohl durch ein manuell ausgeführtes `FW_SafeRelease`, als auch durch eine TwinCAT Vision API-Funktion, die die Variable neu beschreibt, geschehen. Wird daraufhin per Methodenaufruf auf die verbleibende Interface Pointer Variable zugegriffen, führt dies wie zuvor beschrieben zu einem **vollständigen Systemabsturz**.

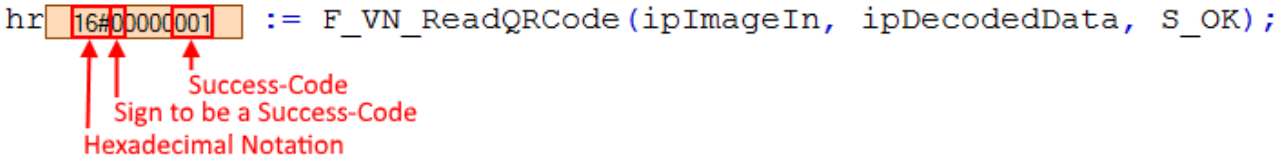
Wenn im Gegensatz dazu Interface Pointer Variablen ohne vorherige Freigabe neu beschrieben werden (z. B. durch eine `:=` Zuweisung), so führt dies zu **Speicher-Lecks**. Dies kann beispielsweise vorkommen, wenn Interface Pointer als Methodenvariablen genutzt und vor Beendigung der Methode nicht freigegeben werden. Ein Indiz für Speicher-Lecks ist im Informationsfenster des AMS Routers zu finden (siehe Router-Speicher [▶ 56]). Wenn der dort als verfügbar markierte Speicher mit der Zeit abnimmt, lässt dies ein Speicher-Leck vermuten.

6.1.1.2 HRESULT

Alle TwinCAT Vision-Funktionen liefern nach ihrer Ausführung ein HRESULT zurück. Sein Wert gibt an, ob die Ausführung erfolgreich war oder fehlgeschlagen ist.

SUCCESS-Codes

Eine erfolgreiche Ausführung ist durch einen POSITVEN Code gekennzeichnet bzw. in der hexadezimalen Schreibweise ist die erste Ziffer zwischen 0 und 7.



Häufige SUCCESS-Codes sind:

Code	Name	Beschreibung
16#000	S_OK	Funktion wurde erfolgreich ausgeführt
16#001	S_FALSE	Funktion wurde erfolgreich ausgeführt, ist aber zu keinem vollständigen Ergebnis gekommen. (Tritt z.B. bei den Code Reading Funktionen auf, wenn kein Code im Bild gefunden wurde.)
16#203	S_PENDING	Asynchrone Methode wurde gestartet, aber es liegt noch kein Ergebnis vor (Tritt z.B. auf beim ersten Aufruf von fbCameraControl.StartAcquisition(), beim 2. Aufruf kommt dann z.B. S_OK zurück)
16#256	S_WATCHDOGTIMEOUT	Funktion wurde vom Watchdog abgebrochen.

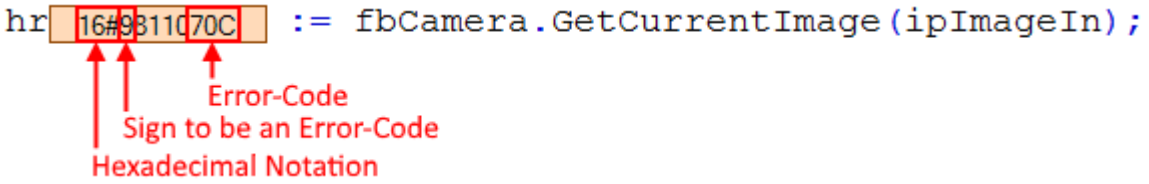
ERROR-Codes

Eine fehlgeschlagene Ausführung ist durch einen NEGATIVEN Code gekennzeichnet, bzw. in der hexadezimalen Schreibweise ist die erste Ziffer >= 8. Die Endziffern des Codes entsprechen wiederum den ADS Return Codes [▶ 2966].

HINWEIS

Error-Code

Die Rückgabe eines Error-Codes bedeutet, dass sämtliche Rückgaben der Funktion invalide sind. Daher darf mit ihnen nicht weitergearbeitet werden.



Gängige ERROR-Codes bei den TwinCAT Vision API-Elementen sind:

Hex	Dec	Name	Beschreibung
16#70A	1802	NOMEMORY	Nicht genügend Speicher
16#70B	1803	INVALIDPARM	Ungültige Parameter-Werte
16#70C	1804	NOTFOUND	Nicht gefunden (Datei, Bild, ...)
16#70E	1806	INCOMPATIBLE	Objekte stimmen nicht überein
16#712	1810	INVALIDSTATE	Die Methode eines FBs wurde in einem unzulässigen Zustand aufgerufen.
16#719	1817	TIMEOUT	Zeitüberschreitung
16#71A	1818	NOINTERFACE	Interface-Abfrage fehlgeschlagen
16#71B	1819	INVALIDINTERFACE	Falsches Interface angefordert.
16#71D	1821	INVALIDOBJID	Object-ID ist ungültig.
16#734	1844	OUTOFRANGE	Außerhalb des gültigen Bereichs.

Die vollständige Liste entnehmen Sie bitte den ADS Return Codes [▶ 2966].

Anwendung

Im Fehlerfall ist das `HRESULT` negativ (Errorcode), ansonsten positiv (Successcode). Verwenden Sie die `FAILED ()` und `SUCCEEDED ()` Funktionen, um dies zu überprüfen.

```
PROGRAM MAIN
VAR
    hr      : HRESULT := S_OK;
END_VAR

IF SUCCEEDED(hr) THEN
    //code
END_IF
```

Eingeschränkte Abfrage

Beachten Sie, dass die obere Abfrage nur auf eine erfolgreiche Ausführung der Funktion, nicht auf ein korrektes oder gewünschtes Ergebnis prüft.

Beispielsweise liefern Code Reading Funktionen [▶ 863] den Successcode S_FALSE (16#001), wenn kein Code im Bild gefunden oder gelesen wird. Um diesen Fall zu erkennen, muss hr direkt mit S_OK oder S_FALSE verglichen werden. Entgegen dessen wird von Vermessungsfunktionen [▶ 1511] NOTFOUND (16#70C) zurückgegeben, wenn das zu vermessende Objekt gänzlich auf dem Bild fehlt.

Die meisten TwinCAT Vision-Funktionen erwarten ein HRESULT als letzten Eingangsparameter und werden nur dann ausgeführt, wenn es sich dabei um einen SUCCESS-Code handelt und somit bestätigt ist, dass es bisher keinen Fehler gegeben hat. Andernfalls wird die jeweilige Funktion nicht ausgeführt, sondern gibt das vorhergehende HRESULT zurück. Dadurch wird sichergestellt, dass bei der Übergabe des HRESULT von Funktion zu Funktion der zuerst aufgetretene Fehlercode ausgegeben wird, statt eines Folgefehlers.

SUCCESS-Codes

SUCCESS-Codes (z. B. S_FALSE) werden nicht weitergegeben und von der nächsten Funktion überschrieben. Bei Bedarf muss explizit der HRESULT-Wert der jeweiligen Funktionen überprüft werden.

Zugriff auf Interface-Methoden

Wenn eine Funktion einen Interface-Pointer erstellen soll, ist das HRESULT der Funktion ein Indiz dafür, ob der Interface-Pointer nun existiert. Zusätzlich muss der Interface-Pointer auf Null-Ungleichheit überprüft werden.

⚠ WARNUNG

Methodenzugriff auf ungültigen Speicherbereich führt zum kompletten Systemabsturz

Bei Interface-Methoden ist eine interne Überprüfung nicht möglich, da die zugehörigen Objekte existieren müssen, um die Methode aufrufen zu können.

Stellen Sie selbst außerhalb des Methodenaufrufs sicher, dass das entsprechende Objekt vorhanden ist (Interface Pointer ungleich 0) und das Programm bisher keinen Fehlercode zurückgeliefert hat.

Dies ist eine sichere Abfrage, um einen neu erstellten Interface-Pointer zu benutzen:

```
IF SUCCEEDED(hr) AND ipImage <> 0 THEN
    hr := ipImage.GetWidth(nWidth);
END_IF
```

Einbindung in eigene Funktionsblöcke

Wenn Sie ein HRESULT in Ihre eigenen Bausteine einbinden wollen, geht das wie folgt:

```
METHOD MyMethod : HRESULT
VAR_INPUT
    hr : HRESULT
END_VAR

IF FAILED(hr) THEN
    MyMethod := hr; //skip the whole method
ELSE
    (*
        code
        code
        code
    *)
    MyMethod := S_OK; // or some error code if something went wrong
END_IF
```

Dies wird im Beispiel Selbstgeschriebene Funktionen [▶ 2917] genauer erklärt.

Extraktion des ADS Return Code

Falls z. B. zur Anzeige in einer HMI der reine ADS Return Code benötigt wird, kann dieser aus dem HRESULT wie folgt in numerischer und textueller Form extrahiert werden:

```
PROGRAM MAIN
VAR
    hr              :   HRESULT;
    nReturnCode     :   DWORD;
    sReturnCode     :   STRING;
END_VAR

nReturnCode := DINT_TO_DWORD(hr) AND 16#FFF;
sReturnCode := DWORD_TO_HEXSTR(nReturnCode, 3, FALSE);
```

Sie benötigen dafür die Bibliothek Tc2_Utilities.

6.1.1.3 Machine Learning

Beim Maschinellen Lernen wird ein Modell für die Lösung einer Aufgabe optimiert. Diesen Prozess bezeichnet man als Training. Mithilfe von trainierten Modellen können komplexe Datenanalysen automatisch durchgeführt werden. Damit lassen sich aufwändige, manuell erstellte Programm-Konstrukte ersetzen. Sowohl das Erstellen, das Training als auch die Ausführung der Machine Learning-Modelle erfolgt in der TwinCAT 3 Echtzeitumgebung, mit Hilfe der bereitgestellten Machine Learning [▶ 1413]-Funktionen. Die damit lösbaren Aufgaben sind vielfältig und lassen sich grob in fünf Aufgabengebiete (Klassifikation, Anomalie-Erkennung, Regression, Clusteranalyse und Merkmals-Transformation) einteilen. Je nach Aufgabengebiet stehen verschiedene Modelltypen zur Verfügung, die jeweils allgemeine oder aufgabenspezifische Vor- und Nachteile haben.

Beim Maschinellen Lernen unterscheidet man zwischen überwachtem (supervised) und unüberwachtem Lernen (unsupervised). Beim überwachten Lernen benötigt man zusätzliche Informationen wie annotierte (z.B. gelabelte) Datensätze, damit das Modell eine Abbildung in bekannte Kategorien (Klassen) oder auf gewünschte Zielgrößen lernen kann. Danach kann das Modell eine Vorhersage (Prädiktion) der Kategorien bzw. Zielgrößen für die unbekannten Daten treffen. Dem gegenüber erfolgt das unüberwachte Lernen ohne Zusatzinformationen nur auf der Basis der Merkmale selbst.

Alle Modelle der Klassifikation und Regression lassen sich generell dem überwachten Lernen zuordnen. Demgegenüber werden alle Modelle zur Anomalie-Erkennung und Clustering dem unüberwachten Lernen zugeordnet. Bei der Merkmalstransformation gibt es beide Möglichkeiten, so basiert das Verfahren zur Linear Discriminant Analysis auf dem überwachten Lernen und die Principal Component Analysis auf dem unüberwachten Lernen.

Die folgende Zuordnung von Modell-Typen zu Aufgabengebieten hilft bei der Auswahl geeigneter Modelle. Es ist zu beachten, dass hier speziell die Handhabung von maschinellem Lernen in TwinCAT Vision aufgezeigt wird und nicht im Allgemeinen.

Modell-Typen	Klassifikation	Anomalie-Erkennung	Regression	Clustering	Merkmals-Transformation
Boost Classifier (Boosting)	x				
K-Means++ (KMPP)		x		x	
K-Nearest Neighbors (KNN)	x	x	x		
Linde–Buzo–Gray (LBG)		x		x	
Linear Discriminant Analysis (LDA)					x
Normal Bayes Classifier (NBC)	x	x			
Principal Component Analysis (PCA)					x
Random Forest (RTrees)	x		x		
Simplified TopoART (STA)	x	x	x	x	
Support Vector Machine (SVM)	x	x	x		
Support Vector Machine-Stochastic Gradient Descent (SVM-SGD)	x				

Es können nur Modelle verwendet werden, die über eine der `F_VN_Create`-Funktionen aus der Machine Learning [▶ 1413] Gruppe erstellt wurden. Die so erzeugten und im Nachgang trainierten Modelle lassen sich über die FB_VN_WriteMlModel [▶ 1658] und FB_VN_ReadMlModel [▶ 1650] Bausteine auf die Festplatte speichern oder laden. Der Import von extern trainierten Modellen, z.B. als ONNX-Dateien, ist nicht vorgesehen.

Grundsätzliches Vorgehen beim maschinellen Lernen:

- Problemstellung analysieren und gewünschtes Ergebnis definieren:
  - Daraus ergibt sich unter anderem das Aufgabengebiet, sowie die verwendbaren Modell-Typen.
- Sammeln von Daten:
  - Trainings-, Validierungs- und Testdaten
  - Die Daten müssen Informationen enthalten, die zum Lösen der Problemstellung geeignet sind.
  - Die Daten müssen alle Varianten und Ausprägungen abdecken, die später vorkommen können.
  - Beim überwachten Lernen müssen die Daten zusätzlich annotiert werden.
- Auswahl und Extraktion von geeigneten Merkmalen:
  - Welche Merkmale sind einmalig, speziell oder liefern relevante Informationen zur Unterscheidung?
  - Redundante Merkmale sollten vermieden werden.
  - Dabei kann eine Merkmalstransformation hilfreich sein.
  - Zum Schluss sollte eine Merkmalsnormalisierung durchgeführt werden, damit die Wertebereiche der unterschiedlichen Merkmale vergleichbar und die Anforderungen des Modells erfüllt werden.
- Bestimmung eines geeigneten Modells und der Hyperparameter (Validieren und Optimieren). (Training des ausgewählten Modells mit den ermittelten Hyperparametern auf den Trainings- und Validierungsdaten).
- Abschließender Test mit zuvor nicht verwendeten, unabhängigen Daten.

Die Daten müssen in Form von Samples vorliegen. Ein Sample ist dabei ein Vektor, der aus Merkmalen besteht. Die Merkmale können sowohl greifbare Größen des assoziierten Bildes (z.B. Farbwerte) oder Konturwerte (z.B. in Form des Schwerpunkts) etc. sein, als auch abstrakte Größen (z.B. das Ergebnis einer PCA), die zwar statistisch das Bild sinnvoll beschreiben, die vom Benutzer jedoch nicht direkt gedeutet

werden können. Für das Training von überwachten Modellen werden für jedes Sample Zusatzinformationen wie z.B. eine Klassenzuordnung benötigt. Eine Ansammlung von mehreren Samples wird als Batch bezeichnet.

Abhängig vom Modelltyp stehen Funktionen zum Sample- und / oder Batch-Training sowie zur Prädiktion zur Verfügung. Bei einigen Modellen kann ein Nachtraining stattfinden, um das Modell während der Laufzeit weiter zu optimieren oder auf neue Gegebenheiten anzupassen.

Merkmalsextraktion

Die Merkmalsextraktion beinhaltet die Extraktion von numerischen Merkmalen aus Bilddaten und ist ein wesentlicher Bestandteil einer Machine Learning-Lösung. Dabei muss man unterscheiden, ob ein Merkmal direkt von einer Funktion bestimmt werden kann, ob eine Kombination von Funktionen notwendig ist oder ob es logische Merkmale eines Objekts sind, die zur Unterscheidung verwendet werden können, wie z.B. eine Anzahl von Löchern. Im Kapitel Funktionen zur Merkmalsextraktion [▶ 155] finden Sie eine beispielhafte Auflistung.

Merkmalsnormalisierung

Die Skalierung von Merkmalen zur Normalisierung wird in erster Linie durchgeführt, um sicherzustellen, dass alle Merkmale gleichermaßen zum Lernprozess des Modells beitragen. Merkmale mit größeren Werten können die Abstandsberechnungen dominieren, wodurch distanzbasierte Modelle empfindlicher auf diese Merkmale reagieren. Um das zu vermeiden und die Anforderungen des Modells zu erfüllen, empfehlen wir generell eine Merkmalsnormalisierung durchzuführen.

Die Funktion F_VN_GetFeatureScales [▶ 1473] stellt drei Skalierungsmöglichkeiten (ETcVnFeatureScalingType [▶ 213]) bereit. Darüber hinaus kann man auch eine eigene Skalierung erstellen und verwenden. Die Anwendung der Skalierung erfolgt mit der Funktion F_VN_FeatureScaling [▶ 1461].

Gleiche Merkmals-Extraktion und -Normalisierung

Es ist zu beachten, dass alle Verarbeitungsschritte sowohl beim Training als auch bei der Anwendung eines Modells durchgeführt werden müssen. Daher müssen die Funktionen an beiden Stellen ausgeführt werden und ggfs. Parameter für die spätere Verwendung gespeichert werden.

6.1.1.3.1 Funktionen zur Merkmalsextraktion

Diese Liste enthält Funktionen, die zur Merkmalsextraktion geeignet sind. Diese Auflistung ist nur beispielhalft zu sehen und hat keinen Anspruch auf Vollständigkeit.

Bildmerkmale

- F_VN_ImageAverage [▶ 1247]
- F_VN_ImageAverageStdDev [▶ 1248]
- F_VN_ImageMoments [▶ 1255]
  - F_VN_HuMomentInvariants [▶ 1569]
- F_VN_CountNonZeroPixels [▶ 1241]
- F_VN_HistogramExp [▶ 1276]
- F_VN_ImageCenterOfMass [▶ 1250]
- F_VN_MaxPixelValue [▶ 1256]
- F_VN_MinPixelValue [▶ 1258]
- F_VN_RegionOrientation [▶ 1260]

Konturmerkmale

- F_VN_ContourArea [▶ 988]
- F_VN_ContourCircularity [▶ 990]
- F_VN_ContourRoundness [▶ 1000]
- F_VN_ContourMoments [▶ 996]

    - F_VN_HuMomentInvariants [▶ 1569]
- F_VN_ContourConvexity [▶ 991]
- F_VN_ContourPerimeter [▶ 999]
- F_VN_ContourOrientation [▶ 997]
- F_VN_ContourCenterOfMass [▶ 989]
- F_VN_ContourEccentricity [▶ 992]
- F_VN_ContourElongation [▶ 993]
- F_VN_ContourExtremePoint [▶ 994]
- F_VN_ContourInertiaRatio [▶ 995]
- F_VN_ConvexHullPoints [▶ 1001]
- F_VN_ConvexityDefects [▶ 1003]
- F_VN_FourierDescriptors [▶ 1012]
- F_VN_EnclosingCircle [▶ 1006]
- F_VN_FitEllipse [▶ 1009]
- F_VN_EnclosingRectangle [▶ 1007]

Bildtextur

- F_VN_Granulometry [▶ 1617]
- F_VN_HaralickFeatures [▶ 1619]

Key Points

- F_VN_KeyPointsAGAST [▶ 1386]
- F_VN_KeyPointsAndDescriptorsAKAZE [▶ 1388]
- F_VN_KeyPointsAndDescriptorsORB [▶ 1393]
- F_VN_KeyPointsFAST [▶ 1395]
- F_VN_KeyPointsGFTT [▶ 1397]
- F_VN_KeyPointsSB [▶ 1401]

6.1.1.4 Watchdogs

Da die Laufzeit vieler Bildverarbeitungsalgorithmen abhängig ist vom Bildinhalt, können Funktionsaufrufe unter ungünstigen Umständen wesentlich länger dauern als üblich.

Ursachen könnten zum Beispiel sich ändernde Lichtverhältnisse, unerwartete Objekte im Bild und andere Faktoren sein, die die Berechnungszeit von Funktionen wie F_VN_FindContours() [▶ 1216] drastisch beeinflussen können. Es können z. B. normalerweise nur 10 Konturen gefunden werden, bei anderer Beleuchtung dann plötzlich 100 oder mehr.

Hierdurch kann es zu Zyklusüberschreitungen kommen, die unbedingt zu vermeiden sind, da sie zu undefinierten Verhalten führen.

Hier sind Watchdogs hilfreich. Watchdogs können einzelne Vision-Funktionen oder ganze Codeabschnitte mit mehreren Funktionen nach einer bestimmten Zeit abbrechen. Einige TwinCAT Vision-Funktionen können während der Ausführung abgebrochen werden und liefern die bis dahin berechneten Teilergebnisse zurück. Falls mehrere Funktionen von einem Watchdog überwacht werden, werden alle verbleibenden Funktionen innerhalb des Watchdog-Bereichs übersprungen, wenn die vorgegebene Zeit abgelaufen ist. Die darauffolgenden Funktionen (nach dem Watchdog-Bereich) werden allerdings wieder ganz normal ausgeführt.

Es stehen zwei Funktionen zur Verfügung, um den Start eines Watchdog-überwachten Bereichs zu definieren, die sich dadurch unterscheiden, ab wann die vorgegebene Zeit anfängt zu zählen:

- F_VN_StartAbsWatchdog [▶ 1020] definiert eine absolute Abbruchzeit bezogen auf den Start des Taskzyklus
- F_VN_StartRelWatchdog [▶ 1022] definiert eine relative Abbruchzeit bezogen auf den Start des Watchdog-Bereichs

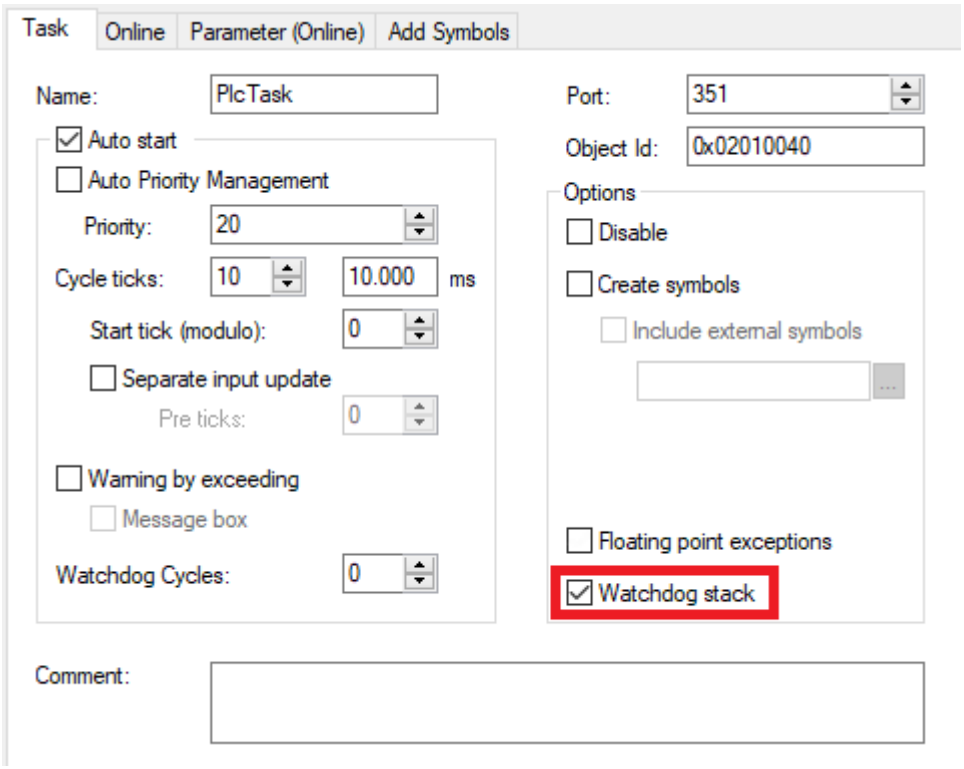
In beiden Fällen wird das Ende des Bereichs durch die Funktion F_VN_StopWatchdog [▶ 1024] definiert, die optional den Bearbeitungsanteil und die benötigte Zeit zurückgibt.

HINWEIS

Watchdog mit relativer Abbruchzeit

Bei der Verwendung der `F_VN_StartRelWatchdog` ist darauf zu achten, dass die Abbruchzeit nicht zu groß gewählt wird, da es sonst weiterhin zu Zyklusüberschreitungen kommen kann. Beachten Sie daher die Zykluszeit der entsprechenden Task, die bis zum Aufruf der Funktion schon vergangene Zeit, sowie die nach `F_VN_StopWatchdog` noch benötigte Zeit bis zum Taskende.

Damit Funktionen durch einen Watchdogs überwacht werden können, muss auf der jeweils ausführenden Task die Option „Watchdog stack“ aktiviert sein.



Beispiel

Im Beispiel unten wird ein Watchdog mit einer Stoppzeit von 10ms relativ zum Aufruf der Watchdog-Funktion gestartet. Angenommen die Zykluszeit beträgt 20ms, beim Aufruf von F_VN_StartRelWatchdog [▶ 1022] sind bereits 4ms im aktuellen Taskzyklus vergangen und der Watchdog wird mit einer Stoppzeit tStop = 10ms bezogen auf die aktuelle Zeit gestartet, dann stoppt der Watchdog die überwachten Funktionen (d. h. jede Funktion, die zwischen dem Start und Stopp des Watchdogs aufgerufen wird), nachdem 14ms des aktuellen Zyklus verstrichen sind.

```
VAR
    ipImage    : ITcVnImage := 0;
    ipContours : ITcVnContainer := 0;

    // watchdog runtime info
    nFunctionsMonitored  : ULINT;
    nFractionProcessed   : UDINT;
    tRest                : DINT;
END_VAR

(* imagine some other functions that use 4ms up to here *)
```

```
hr := F_VN_StartRelWatchdog(10000, hr); // 10ms
    hr := F_VN_Threshold(ipImage, ipImage, 120, 255, TCVN_TT_BINARY, hr);
    hr := F_VN_FindContours(ipImage, ipContours, hr);
hr := F_VN_StopWatchdog(nFunctionsMonitored => nFunctionsMonitored, nFractionProcessed => nFractionP
rocessed, tRest => tRest);
```

Jetzt sind zwei Situationen denkbar:

- Beide Funktionen werden rechtzeitig beendet – dies ist der Normalfall.

Angenommen, F_VN_Threshold [▶ 1363] benötigt 1ms und F_VN_FindContours [▶ 1216] benötigt 5ms. Beim Aufruf von F_VN_StopWatchdog [▶ 1024] verbleiben noch 4ms. Die Watchdog-Laufzeitinformation lautet:

```
nFunctionsMonitored = 2
nFractionProcessed = 100 // in %
tRest = 4000 // in us, equals 4ms
```

- Der Watchdog muss eingreifen.

Szenario: Die Funktion F_VN_Threshold [▶ 1363] ist nicht vom Bildinhalt abhängig (lediglich von der Anzahl Pixel) und benötigt daher wieder nur 1ms, jedoch haben sich die Lichtverhältnisse ungünstig geändert, sodass F_VN_FindContours länger als 9ms benötigen würde. Daher stoppt der Watchdog F_VN_FindContours [▶ 1216], die aber trotzdem die bisher gefundenen Konturen zurückgibt. Die Watchdog-Laufzeitinformation könnte dann folgendermaßen lauten:

```
nFunctionsMonitored = 2
nFractionProcessed = 70
tRest = -50
```

In diesem Fall wurde ein geschätzter Bearbeitungsanteil von 70% berechnet, bis die Funktion abgebrochen wurde. Die Restzeit ist negativ, d. h. die geplante Stoppzeit wurde um 50us überschritten, sodass nach der Funktion F_VN_StopWatchdog [▶ 1024] statt der geplanten 14000us nun 14050us seit dem Start des Taskzyklus vergangen sind. Die Überschreitung geschieht deshalb, weil die bereits berechneten Teilergebnisse der Funktion weiterverwendet werden sollen. Daher kann zum einen der Algorithmus nur an dezidierten Stellen abgebrochen werden, zum anderen müssen die bisherigen Ergebnisse organisiert und zurückgegeben werden. Die maximale Überschreitung hängt dabei generell von der konkreten Funktion und dem Bildinhalt ab. In einem Programm sollte die Abbruchzeit daher immer so gewählt werden, dass noch ein Sicherheitspuffer am Ende des Taskzyklus übrigbleibt.

Überwachte Funktionen

Folgende Funktionen liefern bei einem Watchdog-Abbruch Teilergebnisse:

- F_VN_AdaptiveThreshold(Exp) [▶ 1356]
- F_VN_ConvertColorSpace [▶ 1270]
- F_VN_DetectBlobs(Exp) [▶ 1207]
- F_VN_FindContourHierarchyExp [▶ 1212]
- F_VN_FindContours(Exp) [▶ 1216]
- F_VN_GradeDataMatrixCode(Exp) [▶ 854]
- F_VN_GradeQRCode(Exp) [▶ 860]
- F_VN_HoughCircle(Exp) [▶ 1221]
- F_VN_ImageAbsoluteAverage [▶ 1262]
- F_VN_LocateCircularArc(Exp) [▶ 1520]
- F_VN_LocateEdge(Exp) [▶ 1526]
- F_VN_LocateEdges(Exp) [▶ 1532]
- F_VN_LocateEllipse(Exp) [▶ 1538]
- F_VN_MatchTemplate(Exp) [▶ 1228]

- F_VN_MatchTemplateAndEvaluate(Exp) [▶ 1229]
- F_VN_MeasureAngleBetweenEdges(Exp) [▶ 1544]
- F_VN_MeasureEdgeDistance(Exp) [▶ 1549]
- F_VN_MeasureMinEdgeDistance(Exp) [▶ 1554]
- F_VN_NormalizeImageForDisplay [▶ 1282]
- F_VN_ReadDataMatrixCode(Exp) [▶ 871]
- F_VN_ReadDataMatrixCode(Exp(2)) [▶ 875]
- F_VN_ReferenceColorSimilarity(Exp)_ITcVnColorModel (_ITcVnMlModel) [▶ 1283]
- F_VN_ReferenceColorSimilarity(Exp)_TcVnVector3_LREAL [▶ 1285]
- F_VN_ResizeImage(Exp) [▶ 1188]
- F_VN_RotateImageExp(2) [▶ 1192]
- F_VN_Threshold [▶ 1363]
- F_VN_TrainImageColor(Exp(2)) [▶ 1290]
- F_VN_TrainImageColor(Exp(2))_(ITcVnMlModel) [▶ 1301]
- F_VN_UprightInnerRectangle(Exp) [▶ 1263]
- F_VN_WarpAffine(Exp) [▶ 1194]
- F_VN_WarpPerspective(Exp) [▶ 1199]

Andere Funktionen können während der Ausführung nicht abgebrochen werden, werden allerdings übersprungen, wenn die Zeit des Watchdogs abgelaufen ist.

Beispiele

- Blob Detection mit Watchdog Überwachung [▶ 2867]

6.1.1.5 Bildverarbeitung

In den folgenden Kapiteln sind einige Konzepte der Bilderverarbeitung erklärt, sowohl allgemein, als auch speziell auf TwinCAT Vision bezogen. Sie sind hilfreich, um die Funktionsweise vieler API-Funktionen [▶ 448] zu verstehen.

6.1.1.5.1 Bilder

Die Handhabung von Bildern in TwinCAT Vision geschieht durch Bild-Objekte, die verschiedene Interfaces, wie z. B. das ITcVnImage [▶ 426], implementiert haben können. Auf die tatsächlichen Bilddaten des Bild-Objekts erhält man über einen Interface Pointer Zugriff der dann in der SPS zur Verarbeitung verwendet wird. Die Bilder können dabei aus verschiedenen Pixeltypen und Anzahl an Kanälen bestehen, sowie eine nahezu beliebige Größe haben. Die Bilddaten liegen aufgrund der Verarbeitung aus der Echtzeit und den dynamischen Größen im Router-Speicher. Kamerabilder werden nach dem Empfangen direkt dort abgelegt, Bilder von der Festplatte müssen zunächst über einen File Source oder SPS Baustein eingeladen werden und gelangen auf diesem Weg in den Router Speicher. Die Assistenten im Camera Objekt arbeiten nicht mit Bildern aus dem Router Speicher.

Bild Interface von einem Vision-Gerät abfragen

Häufig werden Bilder durch die Aufnahme mit einer Kamera oder durch das Einladen einer Bilddatei erstellt. Dafür stehen folgende Funktionsblöcke zur Verfügung:

- FB_VN_SimpleCameraControl [▶ 1697]
  Kann mit einer GigE Vision Kamera Instanz oder einer File Source Instanz verbunden werden.
- FB_VN_GevCameraControl [▶ 1675]
  Wird mit einer GigE Vision Kamera Instanz verbunden.

- FB_VN_FileSourceControl [▶ 1664]
  Wird mit einer File Source Instanz verbunden.

Je nach Einstellung des Vision-Geräts kann ein Startup-Vorgang oder ein Bild-Triggern nötig sein. Hier wird nur das Abfragen des Interface Pointers in der SPS dargestellt, über den man Zugriff auf das im Speicher liegende Bild erhält:

```
VAR
    hr: HRESULT;
    fbVisionDevice: FB_VN_SimpleCameraControl;
END_VAR

hr := fbVisionDevice.GetCurrentImage(ipImageIn);
IF SUCCEEDED(hr) AND ipImageIn <> 0 THEN
    // Process image…
END_IF
```

Bildformate

Die beiden meist verwendeten Bildformate sind Mono und RGB. In der Kamera wird das übertragene Bildformat unter dem Register `PixelFormat` eingestellt. Um die anfallende Datenmenge zu reduzieren und so höhere Übertragungsraten realisieren zu können, gibt es auch gepackte oder gefilterte Bildformate, wie z. B. das Bayer Format. Da diese übertragenen Bildformate nicht verwendet werden können bzw. durch den Filter nicht dem Originalbild entsprechen, sollten die Formate direkt nach dem Empfang konvertiert werden.

- Im Fall von gepackten Bildern geschieht die Konvertierung intern beim Aufruf von `GetCurrentImage()` und man erhält direkt das entpackte Bild.
- In allen anderen Fällen, wie beim Bayer Format, sollte das Bild als erstes mit der F_VN_ConvertColorSpace [▶ 1270] gewandelt werden. Danach wird das Eingangsbild nicht mehr benötigt und kann verworfen werden.

Dieser Punkt ist besonders zu beachten, wenn Funktionen verwendet werden, die Bildinformationen interpolieren, wie z. B. F_VN_CompensateLensDistortion [▶ 1121], F_VN_WarpAffine [▶ 1194], F_VN_RotateImageExp [▶ 1191] oder wie die Funktionen zur Kalibration, die durch das Filterbild gestört werden. Daher wird generell empfohlen, das Bild zuerst in die SPS einzulesen, dort zu konvertieren, um alle nachfolgenden Funktionen auf dem Originalbild anzuwenden.

Bilder erstellen

Es gibt Situationen, in denen ein Bild benötigt wird, welches nicht durch eine Kamera aufgenommen oder vom Dateisystem eingeladen wurde. Zum Beispiel soll ein einfarbiges Bild zur Anzeige von Ergebnissen erzeugt werden. Dafür kann unter Angabe von Bildgröße, Pixeltyp und Anzahl der Kanäle die Funktion F_VN_CreateImage [▶ 806] verwendet werden. Der hier gewählte Pixeltyp `TCVN_ET_USINT` entspricht einem üblichen Bild mit 8-Bit pro Kanal. Es ist zu beachten, dass dabei der für das Bild benötigte Speicher nur allokiert, jedoch nicht initialisiert wird. Daher kann das Bild mit der Funktion F_VN_SetPixels [▶ 828] auf eine einheitliche Farbe gesetzt werden.

Maximale Bildgröße

Die maximale Bildgröße, die erstellt, empfangen oder verarbeitet werden kann, ist durch die Anzahl von 2^20 Pixel für Zeilen oder Spalten oder durch die Gesamtgröße von 2^30 Pixeln für ein Bild limitiert, je nachdem, welche Grenze zuerst eintritt.

```
PROGRAM MAIN
VAR
    ipImage     :   ITcVnImage;
    aColorBlack :   TcVnVector4_LREAL := [0, 0, 0];
    hr          :   HRESULT;
END_VAR

hr := F_VN_CreateImage(
    ipImage     :=  ipImage,
    nWidth      :=  640,
    nHeight     :=  480,
    ePixelType  :=  TCVN_ET_USINT,
    nChannelNum :=  3,
    hrPrev      :=  hr
);
hr := F_VN_SetPixels(ipImage, aColorBlack, hr);
```

Alternativ kann ein Bild mit der Funktion F_VN_CreateImageFromArray [▶ 808] auch aus bestehenden Bilddaten erstellt werden.

```
PROGRAM MAIN
VAR
    aImageData  :   ARRAY [0..479, 0..639] OF USINT;
    ipImage     :   ITcVnImage;
    hr          :   HRESULT;
END_VAR

hr := F_VN_CreateImageFromArray(
    pData       :=  ADR(aImageData),
    ipImage     :=  ipImage,
    nWidth      :=  640,
    nHeight     :=  480,
    ePixelType  :=  TCVN_ET_USINT,
    nChannelNum :=  1,
    hrPrev      :=  hr
);
```

Zugriff auf Bilddaten

Mit den Funktionen F_VN_GetPixel [▶ 821] und F_VN_SetPixel [▶ 827] kann auf einzelne Pixel eines Bildes zugegriffen werden. Diese Methode ist einfach anzuwenden. Sie hat allerdings eine relativ hohe Laufzeit, wenn man sie wiederholt auf viele Pixel anwendet. Ich diesem Fall bietet sich vielmehr ein direkter Zugriff auf das Bilddatenarray an. Dazu wird jeweils ein Pointer für eine vollständige Pixelreihe geholt und mit dem Array-Operator auf die einzelnen Pixel zugegriffen:

```
PROGRAM MAIN
VAR
    ipImage     :   ITcVnImage;
    nHeight     :   UDINT;
    nWidth      :   UDINT;
    y           :   UDINT;
    x           :   UDINT;
    // Pointer type must match pixel type of ipImage!
    pRow        :   POINTER TO USINT;
    nPixelValue :   USINT;
END_VAR

// Create ipImage here or set to an existing image

// Determine width and height of image
hr := F_VN_GetImageWidth(ipImage, nWidth, hr);
hr := F_VN_GetImageHeight(ipImage, nHeight, hr);

// Iterate every pixel of the image
FOR y := 0 TO nHeight - 1 DO
    IF hr = S_OK THEN
        hr := ipImage.GetRowPointer(y, ADR(pRow));
        IF hr = S_OK THEN
            FOR x := 0 TO nWidth - 1 DO
                // Access pixel via pRow[x]
                nPixelValue := pRow[x]; // e.g.
                pRow[x] := nPixelValue; // e.g.
            END_FOR
        END_IF
        // It's important to release the pointer. Otherwise there will be memory leaks.
        hr := ipImage.ReleaseRowPointer(ADR(pRow));
    END_IF
END_FOR
```

Anzeigbare Bilder

Um per ADS ein Bild übertragen und anzeigen zu können (z.B. im ADS Image Watch [▶ 141]), muss es zunächst in ein anzeigbares Bild des Typs ITcVnDisplayableImage [▶ 426] konvertiert werden.

Die strikte Unterscheidung zwischen Anzeige und Verarbeitung von Bildern geschieht aus folgendem Grund: Die Bildübertragung per ADS erfolgt asynchron zum SPS-Zyklus. Folglich könnten zeitgleich zur Bildübertragung Bildverarbeitungsfunktionen ausgeführt werden. Um Speicher-Konflikte während der Bildübertragung zu verhindern, ist es daher notwendig, die Speicherbereiche von anzuzeigenden und zu verarbeitenden Bildern zu trennen.

Die beiden Interfaces ITcVnImage [▶ 426] und ITcVnDisplayableImage [▶ 426] schließen einen Bildzugriff zum jeweils ungewünschten Zweck (entweder Verarbeiten oder Übertragen) aus. Zur Konvertierung in ein anzeigbares Bild (ITcVnDisplayableImage [▶ 426]) stehen folgende Funktionen zur Verfügung:

- F_VN_CopyIntoDisplayableImage [▶ 804]
- F_VN_TransformIntoDisplayableImage [▶ 835]
- F_VN_TransformIntoDisplayableImageExp [▶ 836]

Die Funktion F_VN_CopyIntoDisplayableImage [▶ 804] legt eine tiefe Kopie der Bilddaten an und stellt diese als anzeigbares Bild zur Verfügung. Sie sollte immer dann verwendet werden, wenn das anzuzeigende Bild im folgenden Programmablauf weiterverwendet werden wird:

```
// Processing, e.g.:
hr := F_VN_Threshold(ipImage, ipImage, 128, 255, TCVN_TT_BINARY, hr);

// Show intermediate processing step
hr := F_VN_CopyIntoDisplayableImage(ipImage, ipImageDisp, hr);

// Further processing, e.g.:
hr := F_VN_FindContours(ipImage, ipContours, hr);
```

Die Funktion F_VN_TransformIntoDisplayableImage [▶ 835] nutzt hingegen die existierenden Bilddaten und gibt den ursprünglichen Interface-Pointer des Bildes frei. Somit können die Bilddaten nicht mehr verarbeitet werden und stehen gänzlich für die Bildübertragung zur Verfügung. Da keine Kopie der Bilddaten erstellt wird, ist die Ausführungszeit dieser Funktion wesentlich kürzer und es wird kein zusätzlicher Speicher verbraucht. Allerdings ist diese Funktion nur möglich, wenn kein weiterer Pointer auf die Bilddaten existiert. Anderenfalls wird der Fehlercode `E_INCOMPATIBLE (16#70E)` zurückgegeben. Dies kann u.a. dann auftreten, wenn das Ads Communicator Objekt [▶ 133] (z.B. zur Stream-Aufnahme [▶ 110]) aktiviert ist, da dieses für kurze Zeit das Bild intern weiterhin referenziert. Diese Funktion bietet sich daher hauptsächlich für das Ende einer Verarbeitungskette an, um ein z. B. fertig generiertes Ergebnisbild anzuzeigen.

```
// Put last results on image, e.g.:
hr := F_VN_DrawContours(ipContours, -1, ipImageRes, aColorRed, 5, hr);

// Finally transform result image into a displayable image
hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, hr);
```

Die Funktion F_VN_TransformIntoDisplayableImageExp [▶ 836] bietet darüber hinaus die Möglichkeit, nur bei Bedarf eine tiefe Kopie des Originalbildes zu erzeugen. Somit wird, wenn möglich, die ressourcensparende Transform-Variante genutzt und, wenn nötig, die Copy-Variante. Ein entsprechender Anwendungsfall kann auftreten, wenn das AdsCommunitor-Objekt [▶ 132] einer GigE Vision Camera [▶ 72] Instanz aktiviert ist, um Bilder als Stream auf das Dateisystem zu speichern. Dabei existiert zusätzlich zum Pointer in der SPS ein weiterer Bilddaten-Pointer im AdsCommunicator-Objekt. Sofern dieser zum Zeitpunkt der Bildverwendung in der SPS noch nicht freigegeben ist, muss das Bild zum Anzeigen kopiert werden.

```
hr := fbCamera.GetCurrentImage(ipImageIn);
hr := F_VN_CopyImage(ipImageIn, ipImageWork, hr);

// Reliably show original image and try to reuse image data to avoid deep copy
hr := F_VN_TransformIntoDisplayableImageExp(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageInDisp,
    bAllowDeepCopy  :=  TRUE,
    hrPrev          :=  hr
);

// Manipulate ipImageWork during processing here ...
```

Speicherbegrenzung für anzeigbare Bilder

Bilder die in der Entwicklungsumgebung (ADS Image Watch oder Kamera Assistenten) angezeigt werden, liegen im gleichen Arbeitsspeicherbereich der jeweiligen Visual Studio Instanz. Wenn man nun sehr viele oder große Bilder gleichzeitig anschauen möchte, kann es dabei zu Engpässen durch den maximal verfügbaren Speicher pro Instanz kommen.

Da die Bilder zur Anzeige im ADS Image Watch aus der SPS abgerufen werden (ADS-Kommunikation) und dieser Austausch über den Router Speicher erfolgt, muss an dieser Stelle ebenfalls ausreichend Speicher dafür zur Verfügung stehen.

Meta-Informationen

Mit der Funktion F_VN_GetImageInfo [▶ 820] können Meta-Informationen zu einem Bild erlangt werden. Als Ergebnis der Funktion wird eine Struktur vom Typ TcVnImageInfo [▶ 244] zurückgegeben. Die darin enthaltenen Informationen können z. B. genutzt werden, um sicherzustellen, dass ein Bild das erwartete Format hat, um konvertiert zu werden:

```
hr := F_VN_GetImageInfo(ipImageIn, stImageInfo, hr);
IF stImageInfo.stPixelFormat.nChannels = 3 THEN
    hr := F_VN_ConvertColorSpace(ipImageIn, ipImageIn, TCVN_CST_RGB_TO_GRAY, hr);
END_IF
```

6.1.1.5.2 Container

Eine typische Aufgabe in der Bildverarbeitung ist, Objekte im Bild zu finden und deren Form, Position und Größe zu ermitteln. Dabei können die gefundenen Objekte zum Beispiel durch ihre Konturpunkte beschrieben werden. Da die Anzahl der Konturpunkte abhängig von dem konkreten Objekt ist, kann diese stark variieren. Aus diesem Grund werden Container dynamischer Größe verwendet, um diese Konturen zu speichern.

Die Container sind jedoch nicht darauf beschränkt nur Konturpunkte zu speichern, sondern sie können auch eine Vielzahl anderer Datentypen beinhalten. Allerdings können verschiedene Datentypen nicht innerhalb eines Containers gemixt werden. Im Folgenden werden ein paar grundlegende Konzepte im Umgang mit Containern erläutert.

Container erstellen

Ein Container kann mit einem einzigen Funktionsaufruf erstellt werden. Lediglich der Typ (siehe verfügbare Typen [▶ 171]) und die initiale Anzahl von Elementen müssen definiert werden:

```
PROGRAM MAIN
VAR
    hr           : HRESULT;
    ipContainer  : ITcVnContainer;
END_VAR

hr := F_VN_CreateContainer(ipContainer, ContainerType_Vector_REAL, 10, hr);
```

In diesem Beispiel wurde ein neuer Container mit 10 Elementen vom Typ *REAL* erstellt, der intern als C++ Vector organisiert ist. Alternativ kann mit der Funktion F_VN_CopyContainer [▶ 772] auch eine tiefe Kopie eines bestehenden Containers erstellt werden. Dabei wird nicht nur der Pointer auf die Daten kopiert - so wie bei einer Zuweisung an eine andere Variable - sondern die Daten werden in einen neu allokierten Speicherbereich kopiert.

Alle verfügbaren Typen von Containern sind als globale Konstanten [▶ 171] hinterlegt und nach folgendem Muster benannt: Alle Namen beginnen mit „`ContainerType_`“. Darauf folgt der Typ, der sich zusammensetzt aus der Beschreibung des eigentlichen Container-Typs (angelehnt an C++ Container) und dem Element-Typen (z.B. `REAL`, `UDINT`, …). Container können auch selbst wieder Container als Elemente enthalten. Jede Ebene der Container-Struktur wird im Namen durch den jeweiligen Container-Typ, z. B. „`Vector_`“ oder "`String_`", repräsentiert. Daher ergibt sich der Typ-Name eines Containers mit `REAL` Elementen zu `ContainerType_Vector_REAL`.

Elemente hinzufügen

Da die Container in der Lage sind, den zugewiesenen Speicher dynamisch zu erweitern, können einem bestehenden Container auch weitere Elemente hinzugefügt werden. Dazu können die F_VN_AppendToContainer [▶ 566] und F_VN_InsertIntoContainer [▶ 699] Funktionen benutzt werden.

Da es dafür aber intern notwendig sein kann, einen komplett neuen Speicherbereich zu allokieren und die bestehenden Daten zu kopieren, empfiehlt es sich, vorher den maximal notwendigen Speicher zu reservieren. Folglich wird nur einmal neuer Speicher allokiert und das Programm ist insgesamt performanter. Dazu existiert die Funktion F_VN_ReserveContainerMemory [▶ 793], die lediglich Speicher reserviert, die Anzahl der Elemente im Container jedoch nicht verändert.

```
// Reserve memory for 100 elements
hr := F_VN_ReserveContainerMemory(ipContainer, 100, hr);
```

```
// Append the value 1.23 to the end of the container
hr := F_VN_AppendToContainer_REAL(1.23, ipContainer, hr);

// Insert the value 4.56 into the container at index 2 (3rd element)
hr := F_VN_InsertIntoContainer_REAL(4.56, ipContainer, 2, hr);
```

Zugriff auf Elemente

Für den Fall, dass nur auf wenige einzelne Elemente des Containers zugegriffen werden soll, empfiehlt es sich dafür die F_VN_GetAt_... bzw. F_VN_SetAt_... Funktionen zu verwenden. Da in der SPS keine Funktionsnamenüberladung möglich ist, gibt es für jeden unterstützten Containertyp einen eigenen Funktionsnamen. Das erste Element im Container hat dabei den Index 0.

```
PROGRAM MAIN
VAR
    hr           : HRESULT;
    ipContainer  : ITcVnContainer;
    fElement     : REAL;
END_VAR

hr := F_VN_GetAt_REAL(ipContainer, fElement, 0, hr); // Get first element
hr := F_VN_SetAt_REAL(fElement, ipContainer, 2, hr); // Set third element
```

Der Vorteil bei der Verwendung dieser vorgefertigten Funktionen ist, dass nur eine einzelne Zeile Code notwendig ist und intern sämtliche Überprüfungen (z.B. ob der Container-Typ korrekt und der Index nicht zu groß ist) erledigt werden. Möchte man jedoch nacheinander auf alle Elemente des Containers zugreifen, empfiehlt es sich insbesondere bei großen Containern über Iteratoren und das zugehörige Access-Interface auf die Elemente des Containers zuzugreifen, um eine bessere Performance zu erreichen:

```
PROGRAM MAIN
VAR
    hr           :   HRESULT;
    ipContainer  :   ITcVnContainer;
    ipIterator   :   ITcVnForwardIterator;
    ipAccess     :   ITcVnAccess_REAL;
    fElement     :   REAL;
END_VAR

hr := F_VN_GetForwardIterator(ipContainer, ipIterator, hr);
IF SUCCEEDED(hr) AND ipIterator <> 0 THEN
    hr := ipIterator.TcQueryInterface(IID_ITcVnAccess_REAL, ADR(ipAccess));
    IF SUCCEEDED(hr) AND ipAccess <> 0 THEN
        WHILE SUCCEEDED(hr) AND ipIterator.CheckIfEnd() <> S_OK DO
            hr := ipAccess.Get(fElement);
            IF SUCCEEDED(hr) THEN
                fElement := fElement + 1;
                hr := ipAccess.Set(fElement);
            END_IF
            IF SUCCEEDED(hr) THEN
                hr := ipIterator.Increment();
            END_IF
        END_WHILE
    END_IF
    hr := FW_SafeRelease(ADR(ipAccess));
END_IF
hr := FW_SafeRelease(ADR(ipIterator));
```

Falls die Elemente des Containers wiederum Container sind, nutzen Sie stattdessen die folgende Vorgehensweise.

Container von Containern

Da z. B. die Funktion F_VN_FindContours [▶ 1216] nicht nur eine, sondern gleich mehrere Konturen finden kann, können Container selbst auch wieder Container als Elemente beinhalten (wobei aber alle inneren Container Elemente des gleichen Typs beinhalten müssen). Vom Prinzip her lassen sich diese genauso verwenden, wie die oben beschriebenen einfachen Container. So gibt es z. B. die Funktion F_VN_GetAt_ITcVnContainer [▶ 665], um eine aus der SPS nutzbare tiefe Kopie eines beliebigen inneren Containers zu erhalten (einen Interface Pointer auf die originalen Daten zu bekommen ist technisch nicht möglich). Da man auf dieser Ebene aufgrund des allgemeinen ITcVnContainer-Datentyps nicht unterscheiden muss, welche Basiselementtypen in den inneren Containern enthalten sind, vereinfacht sich der Zugriff über Iteratoren, da das spezielle Access Interface nicht mehr notwendig ist:

```
PROGRAM MAIN
VAR
    hr           : HRESULT;
    ipContainer  : ITcVnContainer;
    ipIterator   : ITcVnForwardIterator;
    ipElement    : ITcVnContainer;
END_VAR

hr := F_VN_GetForwardIterator(ipContainer, ipIterator, hr);
IF SUCCEEDED(hr) AND ipIterator <> 0 THEN
    WHILE SUCCEEDED(hr) AND ipIterator.CheckIfEnd() <> S_OK DO
        // ipElement gets a deep copy!
        hr := F_VN_GetContainer(ipIterator, ipElement, hr);
        IF SUCCEEDED(hr) AND ipElement <> 0 THEN
            // 1. extract information or manipulate ipElement
            // 2. write back changes if ipElement was manipulated
            hr := F_VN_SetContainer(ipIterator, ipElement, hr);
        END_IF
        hr := F_VN_IncrementIterator(ipIterator, hr);
    END_WHILE
    // more efficient to release outside loop, GetContainer handles it inside
    hr := FW_SafeRelease(ADR(ipElement));
END_IF
hr := FW_SafeRelease(ADR(ipIterator));
```

Anzeige von Containern

Zurzeit kann der Inhalt von Containern nicht ohne weiteres im Visual Studio Live Debugging angezeigt werden. Stattdessen können die Daten in ein Array exportiert und so betrachtet werden:

```
VAR
    aArray      :   ARRAY [0..9] OF REAL;
    ipContainer :   ITcVnContainer;
    nBufferSize :   ULINT;
END_VAR

hr := F_VN_ExportContainerSize(ipContainer, nBufferSize, hr);
IF nBufferSize = SIZEOF(aArray) THEN
    hr := F_VN_ExportContainer(
        ipContainer :=ipContainer,
        pBuffer     :=ADR(aArray),
        nBufferSize :=nBufferSize,
        hr
    );
END_IF
```

Überschreiben von Containern

Grundsätzlich können Container mit API-Funktionen einfach überschrieben werden. Zurzeit ist ein Überschreiben allerdings nicht möglich, wenn der existierende Container und der zu schreibende Container abweichende Typen haben. Dies wird durch den Fehlercode 70E (INCOMPATIBLE) signalisiert.

6.1.1.5.3 Konturen

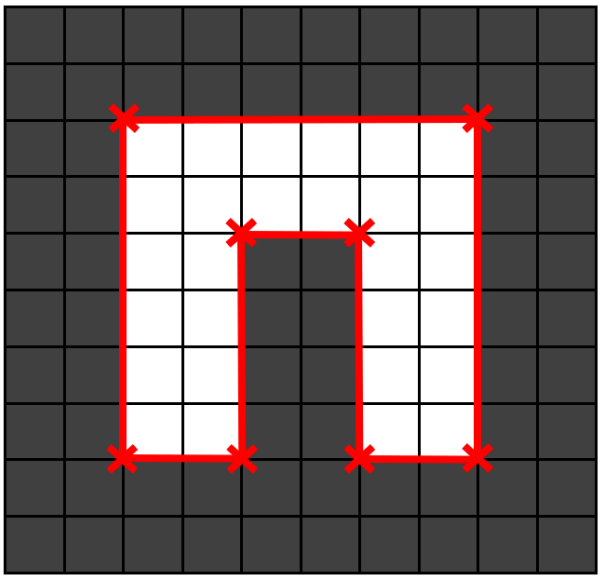
Dieses Kapitel beschreibt die Verwendung von Konturen in TwinCAT Vision.

Was sind Konturen?

Im Allgemeinen ist eine Kontur (auch genannt Umriss) die Abgrenzung eines Objekts von seiner Umgebung, in der Bildverarbeitung klassischerweise zweidimensional. Eine Kontur beschreibt die Form, Größe und Lage dieses Objekts. Im Rahmen der Bildverarbeitung ist damit ein Objekt auf einem Bild [▶ 159] gemeint; daher beschreibt eine Kontur die Umrisslinie des Objekts auf dem Bild.

Wie werden Konturen dargestellt?

Eine Kontur wird als eine Ansammlung von 2D-Punkten dargestellt. Die Punkte beziehen sich meist auf den Ursprung des Bildes, in dem das Objekt abgebildet ist, sofern keine ROI gesetzt wurde. Wenn man alle Punkte dieses Feldes mit geraden Linien verbindet, erhält man eine Zeichnung der Kontur.



Wie werden Konturen in TwinCAT Vision dargestellt?

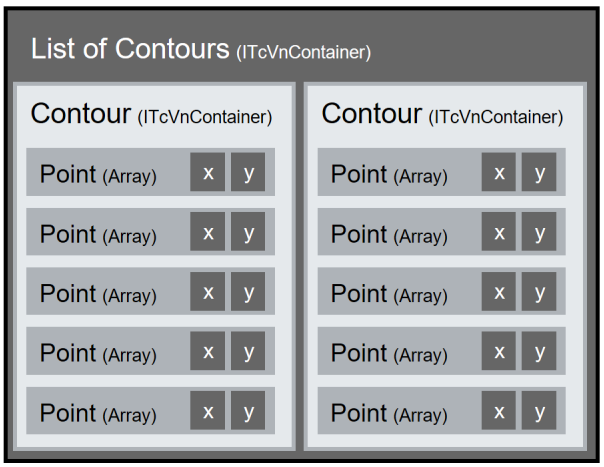
Ein dynamisches Array von 2D-Punkten wird in TwinCAT Vision als Container [▶ 163] (`ITcVnContainer`) von Punkten (`TcVnPoint2_DINT/REAL`) gehandhabt. Ein entsprechender Container hat dann die Typ-Bezeichnung `ContainerType_Vector_TcVnPoint2_DINT/REAL`.

Die Punkte liegen häufig direkt auf dem Pixel-Grid (z. B. bei einer Konturfindung). In diesem Fall werden `DINT`-Elemente zum Speichern der Punkte genutzt. Es gibt jedoch auch Fälle, in denen die Genauigkeit nicht ausreicht, und daher `REAL`s genutzt werden (z. B. bei Vermessungen). Dann liegen die Punkte also nicht genau im Mittelpunkt eines Pixels, sondern irgendwo.

Zeichen-Funktionen können nur Konturen mit ganzzahligen Elementen in ein Bild zeichnen. Im Zweifel muss also der Element-Typ des Containers konvertiert werden.

Felder von Konturen

Häufig kommt es vor, dass mehrere Konturen gleichzeitig gehandhabt werden sollen (z. B. werden bei F_VN_FindContours [▶ 1216] üblicherweise mehrere Konturen gefunden). Das wird technisch derart umgesetzt, indem ein zweidimensionaler Container verwendet wird. Es gibt also ein Array von Konturen bzw. ein Array von Arrays aus 2D-Punkten. Die entsprechende Typ-Bezeichnung ist dann `ContainerType_Vector_Vector_TcVnPoint2_DINT/REAL`.



Was ist Kontur-Approximation?

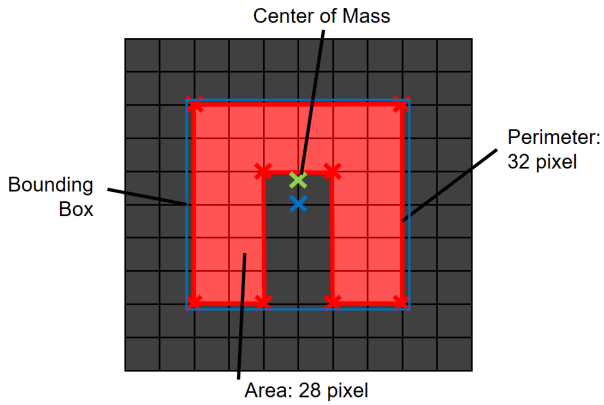
Grundsätzlich kann für die Darstellung eines Objekts immer eine sehr große Anzahl an Punkten verwendet werden. Häufig ist die allerdings nicht nötig, da (näherungsweise) gerade Linien durch ihren Start- und Endpunkt approximiert werden können. Die Punkte dazwischen müssen nicht extra gespeichert werden, weil Start- und Endpunkt für die Definition ausreichen.

Je nach verwendeter Methodik und Geometrie der Objekte, kann die Approximation mehr oder weniger verlustbehaftet sein. Jedenfalls können bei unterschiedlicher Approximation auch die geometrischen Merkmale mit geringfügig anderen Werten berechnet werden.

Geometrische Merkmale

An Konturen, also aus den Arrays von Punkten lassen sich eine Vielzahl von einzelnen Merkmalen berechnen, die die jeweilige Kontur geometrisch beschreiben. Über diese Merkmale kann die Kontur und damit das jeweilige Objekt untersucht und z.B. klassifiziert werden. Zu diesen Merkmalen gehören:

- Umfang
- Fläche
- Mittelpunkte



Geschlossene und offene Konturen

Normalerweise ist eine Kontur geschlossen. In dem Fall werden das erste und das letzte Element des Arrays einfach als verbunden angesehen. Es ist aber auch möglich, eine offene Kontur zur haben. Offene und geschlossene Konturen unterscheiden sich also nicht durch ihre technische Darstellung, sondern über die Information, wie die Daten interpretiert werden sollen. An den relevanten Stellen der API wird entsprechend danach gefragt, ob die jeweilige Kontur offen oder geschlossen verwendet werden soll.

Beispiele

- Object Detection [▶ 2867]
- Contour Analysis [▶ 2857]

6.1.1.5.4 Masken

Dieses Kapitel beschreibt die Bedeutung von **Masken** in TwinCAT Vision.

Was ist eine Maske?

Bezogen auf die Bildverarbeitung meint eine Maske eines Maskenbild, welches zu einem anderen Bild [▶ 159] gehört. Das Maskenbild hat die gleiche Größe und besteht aus einem Grauwert-Kanal. Die Intensitäten des Maskenbildes definieren, welche Pixel des zugehörigen Bildes für bestimmte Arbeitsschritte berücksichtigt werden sollen.

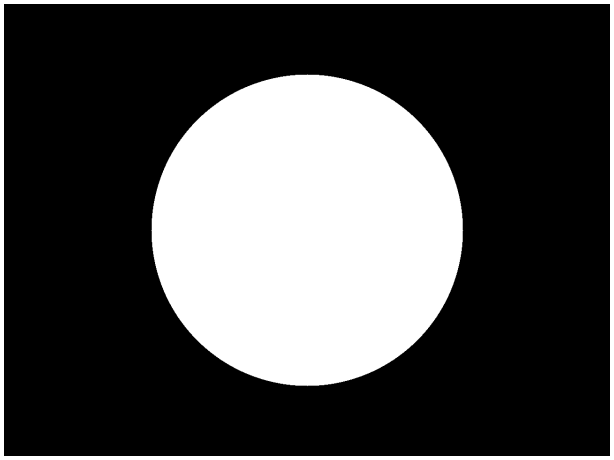
Wofür wird eine Maske benötigt?

Bilder sind ohne Weiteres auf eine rechteckige Form begrenzt. Wenn bei Bildoperation andere Formen als Rechtecke festgelegt werden sollen, dann kann dies mittels einer Maske umgesetzt werden. Die Maske wird dabei durch ein weiteres Bild in gleicher Form und Größe dargestellt, auf dem die zu berücksichtigen Bereiche (Pixel) anders gekennzeichnet werden.

Wie wird eine Maske technisch dargestellt?

Die Intensitäten des Masken-Grauwert-Bildes beschreiben, welche Pixel berücksichtigt werden sollen. Die Interpretation der Intensitäten kann prinzipiell von Funktion zu Funktion abweichen. Häufig werden binär alle 0-wertigen Pixel als nicht zu berücksichtigen und alle anderswertigen Pixel (>=1) als zu berücksichtigen interpretiert. Es sind jedoch auch Fälle denkbar, in denen der Grad der Berücksichtigung feingranularer eingestellt werden kann.

Eine Maske, die den Betrachtungsbereich auf eine kreisförmige Fläche im Bild einschränkt, kann z. B. so aussehen:



Bei Anwendung dieser Maske haben die Pixel in den Ecken eines zugehörigen Bildes keinen Einfluss mehr auf eine folgende Operation.

Beispiel

Beispielhaft wird zu dem Bild `ipImageWork` eine Maske erstellt, die besagt, dass nur ein Kreisbereich des Bildes zu berücksichtigen ist:

```
hr := F_VN_GetImageWidth(ipImageWork, nWidth, hr);
hr := F_VN_GetImageHeight(ipImageWork, nHeight, hr);

hr := F_VN_CreateImage(ipImageMask, nWidth, nHeight, TCVN_ET_USINT, 1, hr);
hr := F_VN_SetPixels(ipImageMask, aColorBlack, hr);
hr := F_VN_DrawCircle(nWidth/2, nHeight/2, MIN(nWidth, nHeight)/2, ipImageMask, aColorWhite, -1,
hr);
```

Daraufhin wird der durchschnittliche Intensitätswert des Bildes innerhalb des Maskenbereichs berechnet:

```
hr := F_VN_ImageAverageExp(ipImageWork, aAverage, ipImageMask, hr);
fAverageIntensityInCircle := aAverage[0];
```

Ohne Maske wäre es nicht ohne weiteres möglich, den Durchschnittswert auf den Kreisbereich einzuschränken, anstatt das vollständige rechteckige Bild zu berücksichtigen.

Das vollständige Beispiel finden Sie hier: Durchschnittsintensität in beliebigen Formen [▶ 2882].

6.1.1.5.5 Region of Interest

Dieses Kapitel beschreibt die Verwendung einer **Region of Interest** in TwinCAT Vision.

Was ist eine Region of Interest?

Eine Region of Interest (ROI) ist ein rechteckiger Bildausschnitt, auf den ein Bild zwischenzeitlich eingeschränkt wird. Dabei wird das Bild an sich nicht verändert, sondern lediglich der zu verarbeitende Ausschnitt verkleinert. Eine ROI kann jederzeit auf das gesamte Bild zurückgesetzt werden.

Wofür wird eine Region of Interest benötigt?

Häufig ist ein vorliegendes Bild größer als der Bereich, der verarbeitet werden soll. In diesem Fall sollte eine ROI gesetzt werden, um folgende Vorteile zu erzielen:

1. Bessere Performance: Die Verarbeitungszeit eines Bildes wird kürzer, wenn der zu verarbeitende Bildausschnitt kleiner ist.
2. Weniger Störungen: Störende Bildobjekte, die sich außerhalb der ROI befinden, werden von der Verarbeitung ausgeschlossen und beeinflussen daher nicht das Ergebnis.

Genauso ist es durch den Einsatz von ROIs möglich, mehrere Bildbereiche separat zu verarbeiten, indem die ROI nacheinander an verschiedene Stellen des Bilds gesetzt wird.

HINWEIS

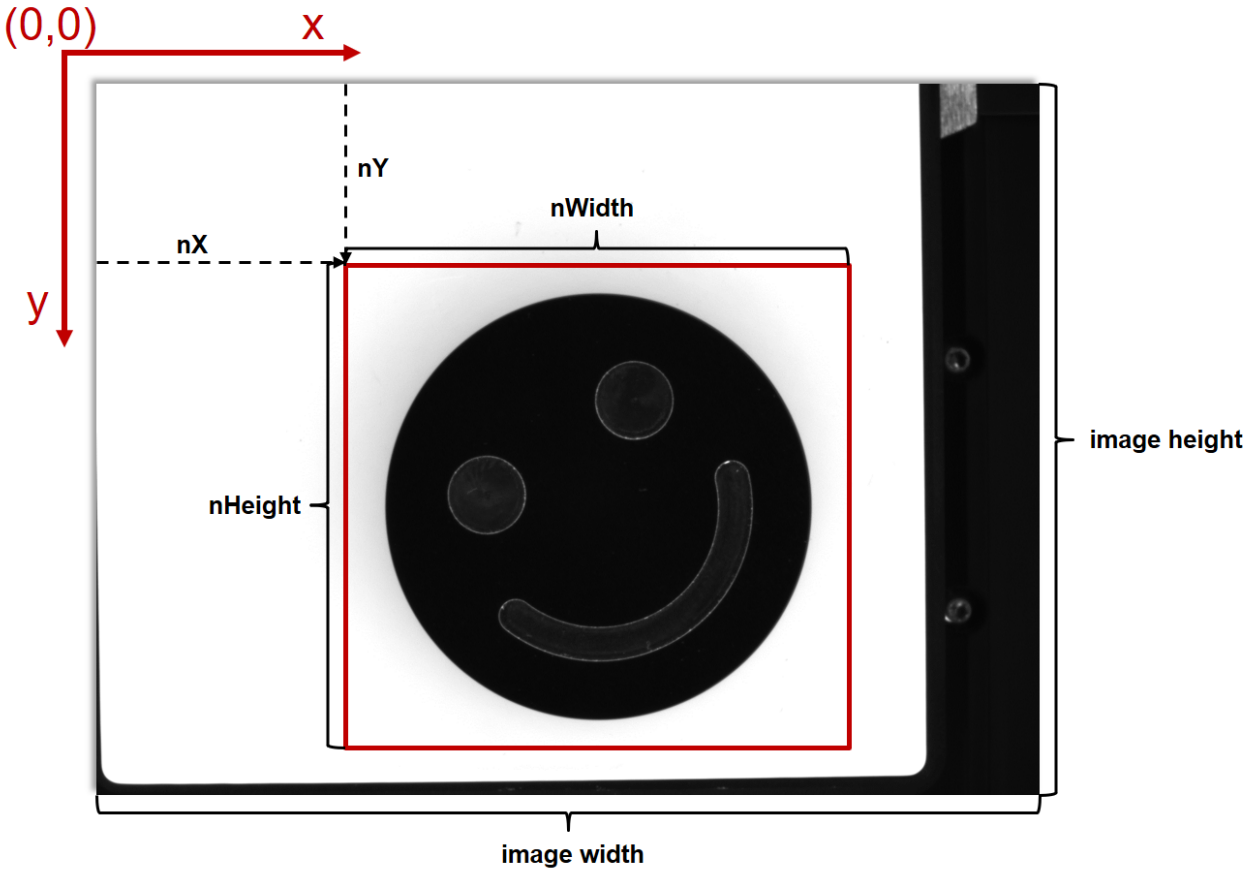
ROI Konfiguration in der Kamera

Wenn Sie eine Kamera als Bildquelle verwenden, setzten Sie am besten dort schon eine ROI um die zu übertragende Datenmenge zu reduzieren und möglichst gering zu halten. Je nach Einstellung und Sensor kann sich dadurch auch die mögliche Bildfrequenz erhöhen. Sehen Sie dazu in den Beispielen zur Kamera-Konfiguration: Region of Interest (ROI) [▶ 2932]

Wie wird eine Region of Interest technisch dargestellt?

Die Information einer gesetzten ROI wird in TwinCAT Vision direkt im jeweiligen Bild-Objekt gespeichert. Das Setzen einer ROI beinhaltet lediglich das Setzen einer Rechteck-Beschreibung. Die Bilddaten außerhalb dieses Bereichs sind jederzeit noch vorhanden, werden aber beim Aufruf von Funktionen nicht berücksichtigt.

Die Rechteck-Beschreibung besteht aus der Position (`nX` und `nY`) der oberen linken Ecke und der Größe (`nWidth` und `nHeight`) des Rechtecks. Es ist zu beachten, dass das Rechteck, nicht über den Bildrand hinausragt (`nWidth ≤ ImageWidth - nX` und `nHeight ≤ ImageHeight - nY`).



Zum Setzen einer ROI werden diese Parameter je nach Funktion direkt oder als Struktur von Typ TcVnRectangle_UDINT [▶ 258] übergeben:

- F_VN_SetRoi [▶ 830]
- F_VN_SetRoi_TcVnRectangle_UDINT [▶ 832]

Um zu überprüfen, ob und welche ROI auf einem Bild gesetzt ist, steht die Funktion F_VN_GetRoi [▶ 823] zur Verfügung. Eine ROI kann auch auf einem Bild gesetzt werden, dass bereits eine ROI hat. Wenn die ROI vergrößert oder auf die gesamte Bildgröße zurückgesetzt werden sollen, kann die Funktion F_VN_ResetRoi [▶ 825] verwendet werden.

6.1.2 Data Types

Zu den Datentypen gehören vier Typen, die jeweils das Präfix „TcVn“ enthalten:

- Alias-Namen [▶ 171] für bestimmte Array-Typen geben Bezug zur Bildverarbeitung
- Arrays [▶ 173] mit einer bestimmten Kombination von Datentypen und definierter Anzahl, die häufig von den Bildverarbeitungsfunktionen verwendet werden.
- Enums [▶ 173] helfen vor allem bei der Parametrierung von Funktionen und enthalten das zusätzliche Präfix „E“
- Strukturen [▶ 238] kapseln mehrere zusammenhängende Informationen (bspw. für geometrische Beschreibungen)

Außerdem sind Konstanten [▶ 171] zur Beschreibung von Container-Typen vorhanden.

6.1.2.1 Aliases

Name	Type
GVCP_REGISTER_ADDRESS	UDINT
TcVnPoint	TcVnPoint2_DINT [▶ 171]
TcVnPoint2_DINT	TcVnVector2_DINT [▶ 173]
TcVnPoint2_LREAL	TcVnVector2_LREAL [▶ 173]
TcVnPoint2_REAL	TcVnVector2_REAL [▶ 173]
TcVnPoint3_LREAL	TcVnVector3_LREAL [▶ 173]
TcVnPoint3_REAL	TcVnVector3_REAL [▶ 173]
TcVnRectangle	TcVnRectangle_DINT [▶ 257]

6.1.2.2 Constants

Die folgenden Konstanten stellen Typen von Containern [▶ 163] als `GUID` dar. Diese werden z. B. bei Funktionen wie F_VN_CreateContainer [▶ 775] benötigt.

Name	Type
ContainerType_Vector_SINT	GUID
ContainerType_Vector_USINT	GUID
ContainerType_Vector_INT	GUID
ContainerType_Vector_UINT	GUID
ContainerType_Vector_DINT	GUID
ContainerType_Vector_UDINT	GUID
ContainerType_Vector_ULINT	GUID
ContainerType_Vector_REAL	GUID
ContainerType_Vector_LREAL	GUID
ContainerType_Vector_ITcVnImage	GUID
ContainerType_Vector_TcVnPoint2_DINT	GUID
ContainerType_Vector_TcVnPoint2_REAL	GUID
ContainerType_Vector_TcVnPoint2_LREAL	GUID
ContainerType_Vector_TcVnPoint3_REAL	GUID
ContainerType_Vector_TcVnPoint3_LREAL	GUID
ContainerType_Vector_Vector_TcVnPoint2_DINT	GUID
ContainerType_Vector_Vector_TcVnPoint2_REAL	GUID
ContainerType_Vector_Vector_TcVnPoint3_REAL	GUID
ContainerType_Vector_Vector_UDINT	GUID
ContainerType_Vector_Vector_ULINT	GUID
ContainerType_Vector_Vector_REAL	GUID
ContainerType_Vector_Vector_LREAL	GUID
ContainerType_Vector_TcVnVector2_SINT	GUID
ContainerType_Vector_TcVnVector2_USINT	GUID
ContainerType_Vector_TcVnVector2_INT	GUID
ContainerType_Vector_TcVnVector2_UINT	GUID
ContainerType_Vector_TcVnVector2_DINT	GUID
ContainerType_Vector_TcVnVector2_REAL	GUID
ContainerType_Vector_TcVnVector3_SINT	GUID
ContainerType_Vector_TcVnVector3_USINT	GUID
ContainerType_Vector_TcVnVector3_INT	GUID
ContainerType_Vector_TcVnVector3_UINT	GUID
ContainerType_Vector_TcVnVector3_REAL	GUID
ContainerType_Vector_TcVnVector4_SINT	GUID
ContainerType_Vector_TcVnVector4_USINT	GUID
ContainerType_Vector_TcVnVector4_INT	GUID
ContainerType_Vector_TcVnVector4_UINT	GUID
ContainerType_Vector_TcVnVector4_DINT	GUID
ContainerType_Vector_TcVnVector4_REAL	GUID
ContainerType_Vector_TcVnVector4_LREAL	GUID
ContainerType_Vector_TcVnRectangle_DINT	GUID
ContainerType_Vector_TcVnRotatedRectangle	GUID
ContainerType_Vector_TcVnKeyPoint	GUID
ContainerType_Vector_TcVnDMatch	GUID
ContainerType_Vector_Vector_TcVnDMatch	GUID
ContainerType_String_SINT	GUID
ContainerType_Vector_String_SINT	GUID

6.1.2.3 Arrays

Name	Type
TcVnArray10_ITcVnImage	array [0..9] of ITcVnImage [▶ 426]
TcVnArray3_Point2_REAL	array [0..2] of TcVnPoint2_REAL [▶ 171]
TcVnArray33_UDINT	array [0..32] of UDINT
TcVnArray4_LREAL	array [0..3] of LREAL
TcVnArray4_Point2_REAL	array [0..3] of TcVnPoint2_REAL [▶ 171]
TcVnArray7_LREAL	array [0..6] of LREAL
TcVnArray8_LREAL	array [0..7] of LREAL
TcVnMatrix2x3_LREAL	array [0..1, 0..2] of LREAL
TcVnMatrix3x3_LREAL	array [0..2, 0..2] of LREAL
TcVnVector2_DINT	array [0..1] of DINT
TcVnVector2_INT	array [0..1] of INT
TcVnVector2_LREAL	array [0..1] of LREAL
TcVnVector2_REAL	array [0..1] of REAL
TcVnVector2_SINT	array [0..1] of SINT
TcVnVector2_UINT	array [0..1] of UINT
TcVnVector2_USINT	array [0..1] of USINT
TcVnVector3_INT	array [0..2] of INT
TcVnVector3_LREAL	array [0..2] of LREAL
TcVnVector3_REAL	array [0..2] of REAL
TcVnVector3_SINT	array [0..2] of SINT
TcVnVector3_UINT	array [0..2] of UINT
TcVnVector3_USINT	array [0..2] of USINT
TcVnVector4_DINT	array [0..3] of DINT
TcVnVector4_INT	array [0..3] of INT
TcVnVector4_LREAL	array [0..3] of LREAL
TcVnVector4_REAL	array [0..3] of REAL
TcVnVector4_SINT	array [0..3] of SINT
TcVnVector4_UINT	array [0..3] of UINT
TcVnVector4_USINT	array [0..3] of USINT

6.1.2.4 Enums

6.1.2.4.1 ETcVn2dCodeSearchStrategy

Offers search strategies for 2d code reading (multiple TCVN_CSS_XXX_INVERTED or multiple TCVN_CSS_XXX_FLIPPED cannot be combined).

Syntax

Definition:

```
TYPE ETcVn2dCodeSearchStrategy :
(
    TCVN_CSS_DEFAULT            := 1,
    TCVN_CSS_ONLY_NOT_INVERTED  := 8,
    TCVN_CSS_FIRST_NOT_INVERTED := 10,
    TCVN_CSS_ONLY_INVERTED      := 12,
    TCVN_CSS_FIRST_INVERTED     := 14,
    TCVN_CSS_ONLY_NOT_FLIPPED   := 64,
    TCVN_CSS_FIRST_NOT_FLIPPED  := 80,
    TCVN_CSS_ONLY_FLIPPED       := 96,
```

```
    TCVN_CSS_FIRST_FLIPPED      := 112
)UDINT;
END_TYPE
```

Values

Name	Description
TCVN_CSS_DEFAULT	The algorithm searches for codes using the default searching strategy, all other flags are ignored (dependant on code type).
TCVN_CSS_ONLY_NOT_INVERTED	The algorithm searches for codes only in the non-inverted image.
TCVN_CSS_FIRST_NOT_INVERTED	The algorithm first searches for codes in the non-inverted and then in the inverted image.
TCVN_CSS_ONLY_INVERTED	The algorithm searches for codes only in the inverted image.
TCVN_CSS_FIRST_INVERTED	The algorithm first searches for codes in the inverted and then in the non-inverted image.
TCVN_CSS_ONLY_NOT_FLIPPED	The algorithm searches for codes only in the non-mirrored image.
TCVN_CSS_FIRST_NOT_FLIPPED	The algorithm first searches for codes in the non-mirrored and then in the mirrored image.
TCVN_CSS_ONLY_FLIPPED	The algorithm searches for codes only in the mirrored image.
TCVN_CSS_FIRST_FLIPPED	The algorithm first searches for codes in the mirrored and then in the non-mirrored image.

Weiterführende Informationen

Dieses Enum bietet Einstellungsmöglichkeiten für die Suchstrategie von Code Reading Funktionen. Weiterführende Informationen finden Sie unter Code Reading Suchstrategien [▶ 864].

Verwandte Funktionen

- F_VN_ReadDataMatrixCodeExp [▶ 874]
- F_VN_ReadQRCodeExp [▶ 888]

6.1.2.4.2 ETcVnAdaptiveThresholdMethod

Offers methods for adaptive threshold.

Syntax

Definition:

```
TYPE ETcVnAdaptiveThresholdMethod :
(
    TCVN_ATM_MEAN     := 0,
    TCVN_ATM_GAUSSIAN := 1
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_ATM_MEAN	The threshold value is calculated as the mean of the nBlockSize x nBlockSize neighborhood of (x,y) minus fConstant.
TCVN_ATM_GAUSSIAN	The threshold value is the weighted sum (cross-correlation with a Gaussian window) of the nBlockSize x nBlockSize neighborhood of (x,y) minus fConstant.

Verwandte Funktionen

- F_VN_AdaptiveThresholdExp [▶ 1358]

6.1.2.4.3 ETcVnBarcodeSearchDirection

Offers search directions for linear barcodes.

Syntax

Definition:

```
TYPE ETcVnBarcodeSearchDirection :
(
    TCVN_BSD_ANY        := 0,
    TCVN_BSD_HORIZONTAL := 1,
    TCVN_BSD_VERTICAL   := 2
)UDINT;
END_TYPE
```

Values

Name	Description
TCVN_BSD_ANY	The algorithm searches for codes in any supported directions.
TCVN_BSD_HORIZONTAL	The algorithm searches for codes in horizontal direction.
TCVN_BSD_VERTICAL	The algorithm searches for codes in vertical direction.

Verwandte Funktionen

- F_VN_ReadBarcodeExp [▶ 867]

6.1.2.4.4 ETcVnBarcodeType

Offers linear barcode types to search for.

Syntax

Definition:

```
TYPE ETcVnBarcodeType :
(
    TCVN_BT_CODABAR        := 4,
    TCVN_BT_CODE39         := 8,
    TCVN_BT_CODE93         := 16,
    TCVN_BT_CODE128        := 32,
    TCVN_BT_EAN8           := 128,
    TCVN_BT_EAN13          := 256,
    TCVN_BT_ITF            := 512,
    TCVN_BT_UPCA           := 32768,
    TCVN_BT_UPCE           := 65536,
    TCVN_BT_ANY            := 99260,
```

```
    TCVN_BT_CODE39EXTENDED := 262144
)UDINT;
END_TYPE
```

Values

Name	Description
TCVN_BT_CODABAR	The algorithm searches for Codabar codes.
TCVN_BT_CODE39	The algorithm searches for Code-39 codes.
TCVN_BT_CODE93	The algorithm searches for Code-93 codes.
TCVN_BT_CODE128	The algorithm searches for Code-128 codes.
TCVN_BT_EAN8	The algorithm searches for EAN-8 codes.
TCVN_BT_EAN13	The algorithm searches for EAN-13 codes.
TCVN_BT_ITF	The algorithm searches for ITF codes.
TCVN_BT_UPCA	The algorithm searches for UPC-A codes.
TCVN_BT_UPCE	The algorithm searches for UPC-E codes.
TCVN_BT_ANY	The algorithm searches for any supported linear barcode. If the type of the code is known, it is recommended to select the specific type directly.
TCVN_BT_CODE39EXTENDED	The algorithm searches for Code-93-Extended codes.

Verwandte Funktionen

- F_VN_ReadBarcode(Exp) [▶ 865]

6.1.2.4.5 ETcVnBlobCombination

Offers multiple blob contours to choose from, which should be returned in a multi-threshold scenario (used in TcVnParamsBlobDetection, which in return is used in F_VN_DetectBlobs).

Syntax

Definition:

```
TYPE ETcVnBlobCombination :
(
    TCVN_BC_SMALLEST         := 0,
    TCVN_BC_LARGEST          := 1,
    TCVN_BC_MIN_THRESHOLD    := 2,
    TCVN_BC_MAX_THRESHOLD    := 3,
    TCVN_BC_MEDIAN_THRESHOLD := 4
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_BC_SMALLEST	Returns the smallest blob contour.
TCVN_BC_LARGEST	Returns the largest blob contour.
TCVN_BC_MIN_THRESHOLD	Returns the blob contour for the minimum threshold.
TCVN_BC_MAX_THRESHOLD	Returns the blob contour for the maximum threshold.
TCVN_BC_MEDIAN_THRESHOLD	Returns the blob contour for the median threshold.

Weiterführende Informationen

Das Enum `ETcVnBlobCombination` wird in der Struktur TcVnParamsBlobDetection [▶ 247] verwendet.

Verwandte Funktionen

- F_VN_CalibrateCamera(Exp) [▶ 1102]

- F_VN_DetectBlobs(Exp) [▶ 1207]

6.1.2.4.6 ETcVnBoostClassifierType

Offers different types of Boost classifiers.

Syntax

Definition:

```
TYPE ETcVnBoostClassifierType :
(
    TCVN_BCT_DISCRETE := 0,
    TCVN_BCT_REAL     := 1,
    TCVN_BCT_LOGIT    := 2,
    TCVN_BCT_GENTLE   := 3
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_BCT_DISCRETE	Discrete AdaBoost
TCVN_BCT_REAL	Real AdaBoost
TCVN_BCT_LOGIT	LogitBoost
TCVN_BCT_GENTLE	Gentle AdaBoost

6.1.2.4.7 ETcVnBorderInterpolationMethod

Offers methods to extrapolate values of non-existing pixels. On the one hand, this is used for filtering functions to enable filtering at the image borders (where the filter mask reaches over the border). On the other hand, this is used to extrapolate the undefined pixels after a geometric image transformation.

Syntax

Definition:

```
TYPE ETcVnBorderInterpolationMethod :
(
    TCVN_BIM_CONSTANT             := 0,
    TCVN_BIM_REPLICATE            := 1,
    TCVN_BIM_REFLECT              := 2,
    TCVN_BIM_WRAP                 := 3,
    TCVN_BIM_REFLECT_101          := 4,
    TCVN_BIM_DEFAULT              := 4,
    TCVN_BIM_TRANSPARENT          := 5,
    TCVN_BIM_ISOLATED_CONSTANT    := 16,
    TCVN_BIM_ISOLATED_REPLICATE   := 17,
    TCVN_BIM_ISOLATED_REFLECT     := 18,
    TCVN_BIM_ISOLATED_WRAP        := 19,
    TCVN_BIM_ISOLATED_REFLECT_101 := 20
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_BIM_CONSTANT	iiiiiii | abcdefgh | iiiiiii (with some specified i)
TCVN_BIM_REPLICATE	aaaaaaa | abcdefgh | hhhhhhh
TCVN_BIM_REFLECT	gfedcba | abcdefgh | hgfedcb
TCVN_BIM_WRAP	bcdefgh | abcdefgh | abcdefg
TCVN_BIM_REFLECT_101	hgfedcb | abcdefgh | gfedcba
TCVN_BIM_DEFAULT	Choose this if you don't know which method to select (same as REFLECT_101).
TCVN_BIM_TRANSPARENT	Corresponding pixels in the destination image will not be modified. Only available for geometric image transformations.
TCVN_BIM_ISOLATED_CONSTANT	Similar to CONSTANT, but ROIs are handled as isolated, which means surrounding image pixels outside the ROI are ignored.
TCVN_BIM_ISOLATED_REPLICATE	Similar to REPLICATE, but ROIs are handled as isolated, which means surrounding image pixels outside the ROI are ignored.
TCVN_BIM_ISOLATED_REFLECT	Similar to REFLECT, but ROIs are handled as isolated, which means surrounding image pixels outside the ROI are ignored.
TCVN_BIM_ISOLATED_WRAP	Similar to WRAP, but ROIs are handled as isolated, which means surrounding image pixels outside the ROI are ignored.
TCVN_BIM_ISOLATED_REFLECT_101	Similar to REFLECT_101, but ROIs are handled as isolated, which means surrounding image pixels outside the ROI are ignored.

Weiterführende Informationen

Die Randinterpolation wird verwendet, um Pixelwerte außerhalb des Bildes zu simulieren. Dies ist z.B. bei der Anwendung eines Filters oder eines morphologischen Operators auf ein Bild notwendig. Wenn beispielsweise ein Filter über das Bild iteriert, muss festgelegt sein, wie sich der Filter an den Bildrändern verhält, wo Teile des Filterkernels außerhalb des Bildes liegen. Dazu werden die nicht vorhandenen Pixel außerhalb des Bildes simuliert, um die Filteroperation dennoch durchführen zu können. Die Art und Weise, wie diese Simulation durchgeführt wird, wird mit dem Enum `ETcVnBorderInterpolationMethod` festgelegt. Häufig ist der Wert `TCVN_BIM_REFLECT` eine gute Wahl; dieser spiegelt die äußeren Bildpixel mit dem Bildrand als Spiegelebene.

Verwandte Funktionen

- F_VN_BilateralFilterExp [▶ 1309]
- F_VN_CustomFilterExp [▶ 1319]
- F_VN_GaussianFilterExp [▶ 1327]
- F_VN_LaplacianFilterExp [▶ 1331]
- F_VN_PadImageBorderExp [▶ 842]
- F_VN_ScharrFilterExp [▶ 1344]
- F_VN_SeparableCustomFilterExp [▶ 1348]
- F_VN_SobelFilterExp [▶ 1352]
- F_VN_WarpAffineExp [▶ 1198]
- F_VN_WarpPerspectiveExp [▶ 1203]

6.1.2.4.8 ETcVnCalibrationPattern

Offers common calibration pattern types.

Syntax

Definition:

```
TYPE ETcVnCalibrationPattern :
(
    TCVN_CP_CIRCLES_SYM         := 0,
    TCVN_CP_CIRCLES_ASYM        := 1,
    TCVN_CP_CIRCLES_ASYM_INDENT := 2
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_CP_CIRCLES_SYM	Symmetric circle pattern
TCVN_CP_CIRCLES_ASYM	Asymmetric circle pattern, i.e. every 2nd row is indented, starting unindented
TCVN_CP_CIRCLES_ASYM_INDENT	Asymmetric circle pattern, i.e. every 2nd row is indented, starting indented

6.1.2.4.9 ETcVnCalibrationPatternOrigin

Offers calibration pattern origins.

Syntax

Definition:

```
TYPE ETcVnCalibrationPatternOrigin :
(
    TCVN_CPO_TOPLEFT      := 0,
    TCVN_CPO_TOPCENTER    := 1,
    TCVN_CPO_TOPRIGHT     := 2,
    TCVN_CPO_CENTERLEFT   := 3,
    TCVN_CPO_CENTER       := 4,
    TCVN_CPO_CENTERRIGHT  := 5,
    TCVN_CPO_BOTTOMLEFT   := 6,
    TCVN_CPO_BOTTOMCENTER := 7,
    TCVN_CPO_BOTTOMRIGHT  := 8
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_CPO_TOPLEFT	Origin is at top left
TCVN_CPO_TOPCENTER	Origin is at top center
TCVN_CPO_TOPRIGHT	Origin is at top right
TCVN_CPO_CENTERLEFT	Origin is at center left
TCVN_CPO_CENTER	Origin is at center
TCVN_CPO_CENTERRIGHT	Origin is at center right
TCVN_CPO_BOTTOMLEFT	Origin is at bottom left
TCVN_CPO_BOTTOMCENTER	Origin is at bottom center
TCVN_CPO_BOTTOMRIGHT	Origin is at bottom right

6.1.2.4.10 ETcVnCameraState

Specifies the state of a camera controller, which controls an attached camera.

Syntax

Definition:

```
TYPE ETcVnCameraState :
(
    TCVN_CS_ERROR             := -1,
    TCVN_CS_INITIAL           := 0,
    TCVN_CS_INITIALIZING      := 1,
    TCVN_CS_INITIALIZED       := 2,
    TCVN_CS_OPENING           := 3,
    TCVN_CS_OPENED            := 4,
    TCVN_CS_STARTACQUISITION  := 5,
    TCVN_CS_ACQUIRING         := 6,
    TCVN_CS_STOPACQUISITION   := 7,
    TCVN_CS_RESETTINGFEATURES := 8,
    TCVN_CS_TRIGGERING        := 9,
    TCVN_CS_CLOSING           := 10
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_CS_ERROR	The camera controller is in an error state (e.g. the connection to the camera was lost).
TCVN_CS_INITIAL	The camera controller is in the initial state and ready to establish the connection to the camera.
TCVN_CS_INITIALIZING	The camera controller is initializing the camera.
TCVN_CS_INITIALIZED	The camera controller has initialized the camera.
TCVN_CS_OPENING	The camera controller is establishing the connection to the camera.
TCVN_CS_OPENED	The connection to the camera has been established and the camera controller is ready to start the image acquisition.
TCVN_CS_STARTACQUISITION	The camera controller is starting the image acquisition.
TCVN_CS_ACQUIRING	The camera is sending images (either streaming or manually triggered, depending on the configuration).
TCVN_CS_STOPACQUISITION	The camera controller is stopping the image acquisition.
TCVN_CS_RESETTINGFEATURES	The camera controller is resetting the features on the camera.
TCVN_CS_TRIGGERING	The camera controller is processing a softwaretrigger.
TCVN_CS_CLOSING	The camera controller is closing the connection to the camera.

Weiterführende Informationen

Die Zustandsautomaten, für die diese Kamerazustände benutzt werden, sind bei den jeweiligen Funktionsblöcken zur Bild-Akquise [▶ 1662] beschrieben:

- FB_VN_GevCameraControl [▶ 1672]
- FB_VN_FileSourceControl [▶ 1664]
- FB_VN_SimpleCameraControl [▶ 1697]

6.1.2.4.11 ETcVnClusteringAlgorithm

Offers clustering algorithms

Syntax

Definition:

```
TYPE ETcVnClusteringAlgorithm :
(
    TCVN_CA_KMEANSPP := 0,
    TCVN_CA_LBG      := 1
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_CA_KMEANSPP	KMeans++ (fixed number of clusters)
TCVN_CA_LBG	LBG variant (dynamic number of clusters)

6.1.2.4.12 ETcVnColorMap

Offers color maps (similar to GNU Octave/MATLAB types).

Syntax

Definition:

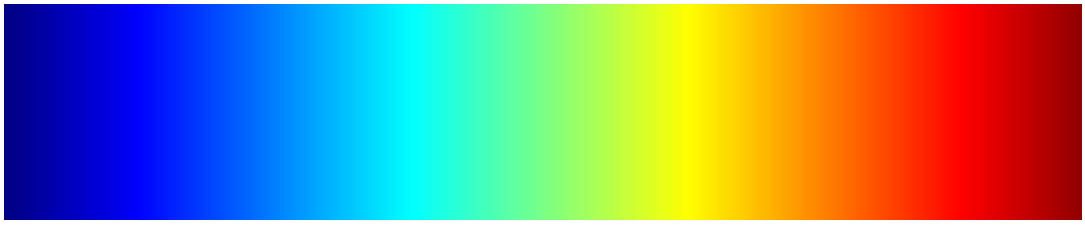
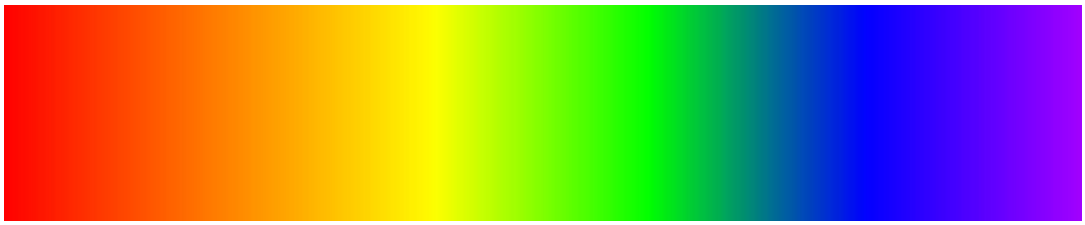
```
TYPE ETcVnColorMap :
(
    TCVN_CM_AUTUMN           := 0,
    TCVN_CM_BONE             := 1,
    TCVN_CM_JET              := 2,
    TCVN_CM_WINTER           := 3,
    TCVN_CM_RAINBOW          := 4,
    TCVN_CM_OCEAN            := 5,
    TCVN_CM_SUMMER           := 6,
    TCVN_CM_SPRING           := 7,
    TCVN_CM_COOL             := 8,
    TCVN_CM_HSV              := 9,
    TCVN_CM_PINK             := 10,
    TCVN_CM_HOT              := 11,
    TCVN_CM_PARULA           := 12,
    TCVN_CM_MAGMA            := 13,
    TCVN_CM_INFERNO          := 14,
    TCVN_CM_PLASMA           := 15,
    TCVN_CM_VIRIDIS          := 16,
    TCVN_CM_CIVIDIS          := 17,
    TCVN_CM_TWILIGHT         := 18,
    TCVN_CM_TWILIGHT_SHIFTED := 19,
    TCVN_CM_TURBO            := 20
)DINT;
END_TYPE
```

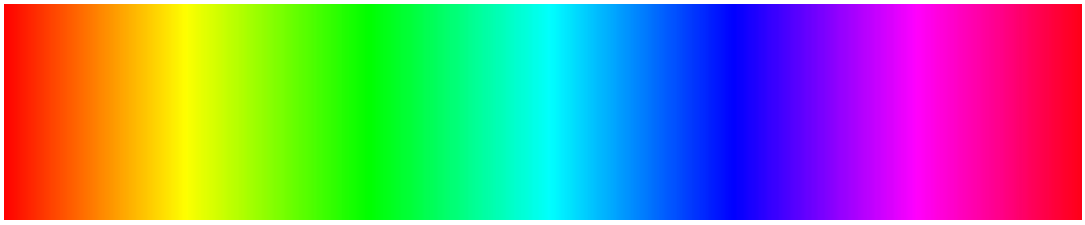
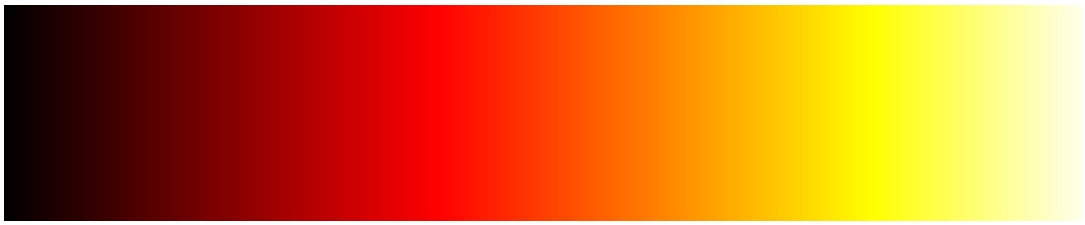
Values

Name	Description
TCVN_CM_AUTUMN	Red - orange - yellow
TCVN_CM_BONE	Black - blueish gray - white
TCVN_CM_JET	Dark blue - green - dark red
TCVN_CM_WINTER	Blue - green
TCVN_CM_RAINBOW	Red - green - purple
TCVN_CM_OCEAN	Black - blue - white
TCVN_CM_SUMMER	Green - yellow
TCVN_CM_SPRING	Pink - yellow
TCVN_CM_COOL	Cyan - magenta
TCVN_CM_HSV	Red - green - blue - red
TCVN_CM_PINK	Black - pink - light yellow - white
TCVN_CM_HOT	Black - red - yellow - white
TCVN_CM_PARULA	Blue - yellow
TCVN_CM_MAGMA	Black - magenta - light yellow
TCVN_CM_INFERNO	Black - magenta - yellow
TCVN_CM_PLASMA	Blue - magenta - yellow
TCVN_CM_VIRIDIS	Purple - yellow
TCVN_CM_CIVIDIS	Blue - gray - yellow
TCVN_CM_TWILIGHT	White - light blue - dark purple - light red - white
TCVN_CM_TWILIGHT_SHIFTED	Dark purple - light blue - white - light red - dark purple
TCVN_CM_TURBO	Dark blue - light green - dark red

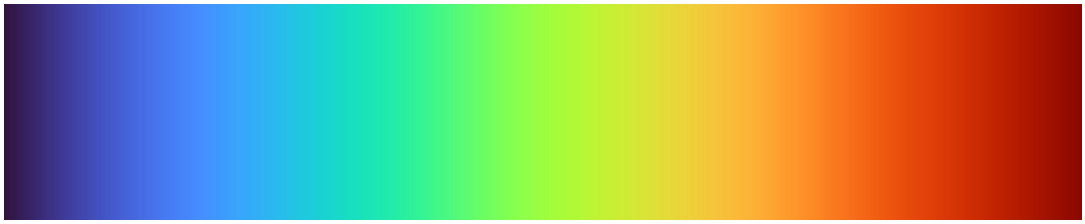
Farbverläufe

Das Enum `ETcVnColorMap` definiert Farbverläufe zum Erstellen von Farbtabellen mit Hilfe der Funktion F_VN_GenerateColorMap [▶ 1270]. Folgende Farbverläufe sind vordefiniert:

NONE	
AUTUMN	
BONE	
JET	
WINTER	
RAINBOW	
OCEAN	

SUMMER	
SPRING	
COOL	
HSV	
Pink	
HOT	
PARULA	

MAGMA	
INFERNO	
PLASMA	
VIRIDIS	
CIVIDIS	
TWILIGHT	
TWILIGHT_SHIFTED	

TURBO	

Verwandte Funktionen

- F_VN_GenerateColorMap [▶ 1270]
- F_VN_GenerateCustomColorMap [▶ 1272]

6.1.2.4.13 ETcVnColorMapSize

Offers color map sizes.

Syntax

Definition:

```
TYPE ETcVnColorMapSize :
(
    TCVN_CMS_256   := 256,
    TCVN_CMS_65536 := 65536
)UDINT;
END_TYPE
```

Values

Name	Description
TCVN_CMS_256	256 colors, used for 8 bit images (TCVN_ET_USINT)
TCVN_CMS_65536	65536 colors, used for 16 bit images (TCVN_ET_UINT)

Verwandte Funktionen

- F_VN_GenerateColorMap [▶ 1270]
- F_VN_GenerateCustomColorMap [▶ 1272]

6.1.2.4.14 ETcVnColorSpaceTransform

Offers color space transformations.

Syntax

Definition:

```
TYPE ETcVnColorSpaceTransform :
(
    TCVN_CST_BGR_TO_BGRA                := 0,
    TCVN_CST_RGB_TO_RGBA                := 0,
    TCVN_CST_BGRA_TO_BGR                := 1,
    TCVN_CST_RGBA_TO_RGB                := 1,
    TCVN_CST_BGR_TO_RGBA                := 2,
    TCVN_CST_RGB_TO_BGRA                := 2,
    TCVN_CST_BGRA_TO_RGB                := 3,
    TCVN_CST_RGBA_TO_BGR                := 3,
    TCVN_CST_BGR_TO_RGB                 := 4,
    TCVN_CST_RGB_TO_BGR                 := 4,
    TCVN_CST_BGRA_TO_RGBA               := 5,
    TCVN_CST_RGBA_TO_BGRA               := 5,
    TCVN_CST_BGR_TO_GRAY                := 6,
    TCVN_CST_RGB_TO_GRAY                := 7,
    TCVN_CST_GRAY_TO_BGR                := 8,
```

```
    TCVN_CST_GRAY_TO_RGB                := 8,
    TCVN_CST_GRAY_TO_BGRA               := 9,
    TCVN_CST_GRAY_TO_RGBA               := 9,
    TCVN_CST_BGRA_TO_GRAY               := 10,
    TCVN_CST_RGBA_TO_GRAY               := 11,
    TCVN_CST_RGB_TO_BGR_565             := 12,
    TCVN_CST_BGR_TO_BGR_565             := 13,
    TCVN_CST_BGR_565_TO_RGB             := 14,
    TCVN_CST_BGR_565_TO_BGR             := 15,
    TCVN_CST_RGBA_TO_BGR_565            := 16,
    TCVN_CST_BGRA_TO_BGR_565            := 17,
    TCVN_CST_BGR_565_TO_RGBA            := 18,
    TCVN_CST_BGR_565_TO_BGRA            := 19,
    TCVN_CST_GRAY_TO_BGR_565            := 20,
    TCVN_CST_BGR_565_TO_GRAY            := 21,
    TCVN_CST_RGB_TO_BGR_555             := 22,
    TCVN_CST_BGR_TO_BGR_555             := 23,
    TCVN_CST_BGR_555_TO_RGB             := 24,
    TCVN_CST_BGR_555_TO_BGR             := 25,
    TCVN_CST_RGBA_TO_BGR_555            := 26,
    TCVN_CST_BGRA_TO_BGR_555            := 27,
    TCVN_CST_BGR_555_TO_RGBA            := 28,
    TCVN_CST_BGR_555_TO_BGRA            := 29,
    TCVN_CST_GRAY_TO_BGR_555            := 30,
    TCVN_CST_BGR_555_TO_GRAY            := 31,
    TCVN_CST_BGR_TO_XYZ                 := 32,
    TCVN_CST_RGB_TO_XYZ                 := 33,
    TCVN_CST_XYZ_TO_BGR                 := 34,
    TCVN_CST_XYZ_TO_RGB                 := 35,
    TCVN_CST_BGR_TO_YCRCB               := 36,
    TCVN_CST_RGB_TO_YCRCB               := 37,
    TCVN_CST_YCRCB_TO_BGR               := 38,
    TCVN_CST_YCRCB_TO_RGB               := 39,
    TCVN_CST_BGR_TO_HSV                 := 40,
    TCVN_CST_RGB_TO_HSV                 := 41,
    TCVN_CST_BGR_TO_LAB                 := 44,
    TCVN_CST_RGB_TO_LAB                 := 45,
    TCVN_CST_BAYER_RG_TO_BGR            := 46,
    TCVN_CST_BAYER_BG_TO_RGB            := 46,
    TCVN_CST_BAYER_GR_TO_BGR            := 47,
    TCVN_CST_BAYER_GB_TO_RGB            := 47,
    TCVN_CST_BAYER_BG_TO_BGR            := 48,
    TCVN_CST_BAYER_RG_TO_RGB            := 48,
    TCVN_CST_BAYER_GB_TO_BGR            := 49,
    TCVN_CST_BAYER_GR_TO_RGB            := 49,
    TCVN_CST_BGR_TO_LUV                 := 50,
    TCVN_CST_RGB_TO_LUV                 := 51,
    TCVN_CST_BGR_TO_HLS                 := 52,
    TCVN_CST_RGB_TO_HLS                 := 53,
    TCVN_CST_HSV_TO_BGR                 := 54,
    TCVN_CST_HSV_TO_RGB                 := 55,
    TCVN_CST_LAB_TO_BGR                 := 56,
    TCVN_CST_LAB_TO_RGB                 := 57,
    TCVN_CST_LUV_TO_BGR                 := 58,
    TCVN_CST_LUV_TO_RGB                 := 59,
    TCVN_CST_HLS_TO_BGR                 := 60,
    TCVN_CST_HLS_TO_RGB                 := 61,
    TCVN_CST_BAYER_RG_TO_BGR_VNG        := 62,
    TCVN_CST_BAYER_GR_TO_BGR_VNG        := 63,
    TCVN_CST_BAYER_BG_TO_BGR_VNG        := 64,
    TCVN_CST_BAYER_GB_TO_BGR_VNG        := 65,
    TCVN_CST_BAYER_RG_TO_RGB_VNG        := 64,
    TCVN_CST_BAYER_GR_TO_RGB_VNG        := 65,
    TCVN_CST_BAYER_BG_TO_RGB_VNG        := 62,
    TCVN_CST_BAYER_GB_TO_RGB_VNG        := 63,
    TCVN_CST_BGR_TO_HSV_FULL            := 66,
    TCVN_CST_RGB_TO_HSV_FULL            := 67,
    TCVN_CST_BGR_TO_HLS_FULL            := 68,
    TCVN_CST_RGB_TO_HLS_FULL            := 69,
    TCVN_CST_HSV_TO_BGR_FULL            := 70,
    TCVN_CST_HSV_TO_RGB_FULL            := 71,
    TCVN_CST_HLS_TO_BGR_FULL            := 72,
    TCVN_CST_HLS_TO_RGB_FULL            := 73,
    TCVN_CST_LBGR_TO_LAB                := 74,
    TCVN_CST_LRGB_TO_LAB                := 75,
    TCVN_CST_LBGR_TO_LUV                := 76,
    TCVN_CST_LRGB_TO_LUV                := 77,
    TCVN_CST_LAB_TO_LBGR                := 78,
    TCVN_CST_LAB_TO_LRGB                := 79,
    TCVN_CST_LUV_TO_LBGR                := 80,
```

```
    TCVN_CST_LUV_TO_LRGB                := 81,
    TCVN_CST_BGR_TO_YUV                 := 82,
    TCVN_CST_RGB_TO_YUV                 := 83,
    TCVN_CST_YUV_TO_BGR                 := 84,
    TCVN_CST_YUV_TO_RGB                 := 85,
    TCVN_CST_BAYER_RG_TO_GRAY           := 86,
    TCVN_CST_BAYER_GR_TO_GRAY           := 87,
    TCVN_CST_BAYER_BG_TO_GRAY           := 88,
    TCVN_CST_BAYER_GB_TO_GRAY           := 89,
    TCVN_CST_YUV_420_NV12_TO_RGB        := 90,
    TCVN_CST_YUV_420_NV12_TO_BGR        := 91,
    TCVN_CST_YUV_420_NV21_TO_RGB        := 92,
    TCVN_CST_YUV_420_NV21_TO_BGR        := 93,
    TCVN_CST_YUV_420_SP_TO_RGB          := 92,
    TCVN_CST_YUV_420_SP_TO_BGR          := 93,
    TCVN_CST_YUV_420_NV12_TO_RGBA       := 94,
    TCVN_CST_YUV_420_NV12_TO_BGRA       := 95,
    TCVN_CST_YUV_420_NV21_TO_RGBA       := 96,
    TCVN_CST_YUV_420_NV21_TO_BGRA       := 97,
    TCVN_CST_YUV_420_SP_TO_RGBA         := 96,
    TCVN_CST_YUV_420_SP_TO_BGRA         := 97,
    TCVN_CST_YUV_420_YV12_TO_RGB        := 98,
    TCVN_CST_YUV_420_YV12_TO_BGR        := 99,
    TCVN_CST_YUV_420_IYUV_TO_RGB        := 100,
    TCVN_CST_YUV_420_IYUV_TO_BGR        := 101,
    TCVN_CST_YUV_420_I420_TO_RGB        := 100,
    TCVN_CST_YUV_420_I420_TO_BGR        := 101,
    TCVN_CST_YUV_420_P_TO_RGB           := 98,
    TCVN_CST_YUV_420_P_TO_BGR           := 99,
    TCVN_CST_YUV_420_YV12_TO_RGBA       := 102,
    TCVN_CST_YUV_420_YV12_TO_BGRA       := 103,
    TCVN_CST_YUV_420_IYUV_TO_RGBA       := 104,
    TCVN_CST_YUV_420_IYUV_TO_BGRA       := 105,
    TCVN_CST_YUV_420_I420_TO_RGBA       := 104,
    TCVN_CST_YUV_420_I420_TO_BGRA       := 105,
    TCVN_CST_YUV_420_P_TO_RGBA          := 102,
    TCVN_CST_YUV_420_P_TO_BGRA          := 103,
    TCVN_CST_YUV_420_TO_GRAY            := 106,
    TCVN_CST_YUV_420_NV21_TO_GRAY       := 106,
    TCVN_CST_YUV_420_NV12_TO_GRAY       := 106,
    TCVN_CST_YUV_420_YV12_TO_GRAY       := 106,
    TCVN_CST_YUV_420_IYUV_TO_GRAY       := 106,
    TCVN_CST_YUV_420_I420_TO_GRAY       := 106,
    TCVN_CST_YUV_420_SP_TO_GRAY         := 106,
    TCVN_CST_YUV_420_P_TO_GRAY          := 106,
    TCVN_CST_YUV_422_UYVY_TO_RGB        := 107,
    TCVN_CST_YUV_422_UYVY_TO_BGR        := 108,
    TCVN_CST_YUV_422_Y422_TO_RGB        := 107,
    TCVN_CST_YUV_422_Y422_TO_BGR        := 108,
    TCVN_CST_YUV_422_UYNV_TO_RGB        := 107,
    TCVN_CST_YUV_422_UYNV_TO_BGR        := 108,
    TCVN_CST_YUV_422_UYVY_TO_RGBA       := 111,
    TCVN_CST_YUV_422_UYVY_TO_BGRA       := 112,
    TCVN_CST_YUV_422_Y422_TO_RGBA       := 111,
    TCVN_CST_YUV_422_Y422_TO_BGRA       := 112,
    TCVN_CST_YUV_422_UYNV_TO_RGBA       := 111,
    TCVN_CST_YUV_422_UYNV_TO_BGRA       := 112,
    TCVN_CST_YUV_422_YUY2_TO_RGB        := 115,
    TCVN_CST_YUV_422_YUY2_TO_BGR        := 116,
    TCVN_CST_YUV_422_YVYU_TO_RGB        := 117,
    TCVN_CST_YUV_422_YVYU_TO_BGR        := 118,
    TCVN_CST_YUV_422_YUYV_TO_RGB        := 115,
    TCVN_CST_YUV_422_YUYV_TO_BGR        := 116,
    TCVN_CST_YUV_422_YUNV_TO_RGB        := 115,
    TCVN_CST_YUV_422_YUNV_TO_BGR        := 116,
    TCVN_CST_YUV_422_YUY2_TO_RGBA       := 119,
    TCVN_CST_YUV_422_YUY2_TO_BGRA       := 120,
    TCVN_CST_YUV_422_YVYU_TO_RGBA       := 121,
    TCVN_CST_YUV_422_YVYU_TO_BGRA       := 122,
    TCVN_CST_YUV_422_YUYV_TO_RGBA       := 119,
    TCVN_CST_YUV_422_YUYV_TO_BGRA       := 120,
    TCVN_CST_YUV_422_YUNV_TO_RGBA       := 119,
    TCVN_CST_YUV_422_YUNV_TO_BGRA       := 120,
    TCVN_CST_YUV_422_UYVY_TO_GRAY       := 123,
    TCVN_CST_YUV_422_YUY2_TO_GRAY       := 124,
    TCVN_CST_YUV_422_Y422_TO_GRAY       := 123,
    TCVN_CST_YUV_422_UYNV_TO_GRAY       := 123,
    TCVN_CST_YUV_422_YVYU_TO_GRAY       := 124,
    TCVN_CST_YUV_422_YUYV_TO_GRAY       := 124,
    TCVN_CST_YUV_422_YUNV_TO_GRAY       := 124,
```

```
    TCVN_CST_RGBA_TO_PREMULTIPLICATED_RGBA := 125,
    TCVN_CST_PREMULTIPLICATED_RGBA_TO_RGBA := 126,
    TCVN_CST_RGB_TO_YUV_420_I420           := 127,
    TCVN_CST_BGR_TO_YUV_420_I420           := 128,
    TCVN_CST_RGB_TO_YUV_420_IYUV           := 127,
    TCVN_CST_BGR_TO_YUV_420_IYUV           := 128,
    TCVN_CST_RGBA_TO_YUV_420_I420          := 129,
    TCVN_CST_BGRA_TO_YUV_420_I420          := 130,
    TCVN_CST_RGBA_TO_YUV_420_IYUV          := 129,
    TCVN_CST_BGRA_TO_YUV_420_IYUV          := 130,
    TCVN_CST_RGB_TO_YUV_420_YV12           := 131,
    TCVN_CST_BGR_TO_YUV_420_YV12           := 132,
    TCVN_CST_RGBA_TO_YUV_420_YV12          := 133,
    TCVN_CST_BGRA_TO_YUV_420_YV12          := 134,
    TCVN_CST_BAYER_RG_TO_BGR_EA            := 135,
    TCVN_CST_BAYER_GR_TO_BGR_EA            := 136,
    TCVN_CST_BAYER_BG_TO_BGR_EA            := 137,
    TCVN_CST_BAYER_GB_TO_BGR_EA            := 138,
    TCVN_CST_BAYER_RG_TO_RGB_EA            := 137,
    TCVN_CST_BAYER_GR_TO_RGB_EA            := 138,
    TCVN_CST_BAYER_BG_TO_RGB_EA            := 135,
    TCVN_CST_BAYER_GB_TO_RGB_EA            := 136,
    TCVN_CST_BAYER_RG_TO_BGRA              := 139,
    TCVN_CST_BAYER_GR_TO_BGRA              := 140,
    TCVN_CST_BAYER_BG_TO_BGRA              := 141,
    TCVN_CST_BAYER_GB_TO_BGRA              := 142,
    TCVN_CST_BAYER_RG_TO_RGBA              := 141,
    TCVN_CST_BAYER_GR_TO_RGBA              := 142,
    TCVN_CST_BAYER_BG_TO_RGBA              := 139,
    TCVN_CST_BAYER_GB_TO_RGBA              := 140,
    TCVN_CST_MAX                           := 143
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_CST_BGR_TO_BGRA	Transform the color space from BGR to BGRA.
TCVN_CST_RGB_TO_RGBA	Transform the color space from RGB to RGBA.
TCVN_CST_BGRA_TO_BGR	Transform the color space from BGRA to BGR.
TCVN_CST_RGBA_TO_RGB	Transform the color space from RGBA to RGB.
TCVN_CST_BGR_TO_RGBA	Transform the color space from BGR to RGBA.
TCVN_CST_RGB_TO_BGRA	Transform the color space from RGB to BGRA.
TCVN_CST_BGRA_TO_RGB	Transform the color space from BGRA to RGB.
TCVN_CST_RGBA_TO_BGR	Transform the color space from RGBA to BGR.
TCVN_CST_BGR_TO_RGB	Transform the color space from BGR to RGB.
TCVN_CST_RGB_TO_BGR	Transform the color space from RGB to BGR.
TCVN_CST_BGRA_TO_RGBA	Transform the color space from BGRA to RGBA.
TCVN_CST_RGBA_TO_BGRA	Transform the color space from RGBA to BGRA.
TCVN_CST_BGR_TO_GRAY	Transform the color space from BGR to Gray.
TCVN_CST_RGB_TO_GRAY	Transform the color space from RGB to Gray.
TCVN_CST_GRAY_TO_BGR	Transform the color space from Gray to BGR.
TCVN_CST_GRAY_TO_RGB	Transform the color space from Gray to RGB.
TCVN_CST_GRAY_TO_BGRA	Transform the color space from Gray to BGRA.
TCVN_CST_GRAY_TO_RGBA	Transform the color space from Gray to RGBA.
TCVN_CST_BGRA_TO_GRAY	Transform the color space from BGRA to Gray.
TCVN_CST_RGBA_TO_GRAY	Transform the color space from RGBA to Gray.
TCVN_CST_RGB_TO_BGR_565	Transform the color space from RGB(24 bit 888) to BGR(16 bit 565).
TCVN_CST_BGR_TO_BGR_565	Transform the color space from BGR(24 bit 888) to BGR(16 bit 565).
TCVN_CST_BGR_565_TO_RGB	Transform the color space from BGR(16 bit 565) to RGB(24 bit 888).
TCVN_CST_BGR_565_TO_BGR	Transform the color space from BGR(16 bit 565) to BGR(24 bit 888).
TCVN_CST_RGBA_TO_BGR_565	Transform the color space from RGBA(32 bit 8888) to BGR(16 bit 565).
TCVN_CST_BGRA_TO_BGR_565	Transform the color space from BGRA(32 bit 8888) to BGR(16 bit 565).
TCVN_CST_BGR_565_TO_RGBA	Transform the color space from BGR(16 bit 565) to RGBA(32 bit 8888).
TCVN_CST_BGR_565_TO_BGRA	Transform the color space from BGR(16 bit 565) to BGRA(32 bit 8888).
TCVN_CST_GRAY_TO_BGR_565	Transform the color space from Gray to BGR(16 bit 565).
TCVN_CST_BGR_565_TO_GRAY	Transform the color space from BGR(16 bit 565) to Gray.
TCVN_CST_RGB_TO_BGR_555	Transform the color space from RGB(24 bit 888) to BGR(16 bit 555).
TCVN_CST_BGR_TO_BGR_555	Transform the color space from BGR(24 bit 888) to BGR(16 bit 555).
TCVN_CST_BGR_555_TO_RGB	Transform the color space from BGR(16 bit 555) to RGB(24 bit 888).
TCVN_CST_BGR_555_TO_BGR	Transform the color space from BGR(16 bit 555) to BGR(24 bit 888).
TCVN_CST_RGBA_TO_BGR_555	Transform the color space from RGBA(32 bit 8888) to BGR(16 bit 555).
TCVN_CST_BGRA_TO_BGR_555	Transform the color space from BGRA(32 bit 8888) to BGR(16 bit 555).

Name	Description
TCVN_CST_BGR_555_TO_RGBA	Transform the color space from BGR(16 bit 555) to RGBA(32 bit 8888).
TCVN_CST_BGR_555_TO_BGRA	Transform the color space from BGR(16 bit 555) to BGRA(32 bit 8888).
TCVN_CST_GRAY_TO_BGR_555	Transform the color space from Gray to BGR(16 bit 555).
TCVN_CST_BGR_555_TO_GRAY	Transform the color space from BGR(16 bit 555) to Gray.
TCVN_CST_BGR_TO_XYZ	Transform the color space from BGR to CIE XYZ (scaled to the full value range of the image).
TCVN_CST_RGB_TO_XYZ	Transform the color space from RGB to CIE XYZ (scaled to the full value range of the image).
TCVN_CST_XYZ_TO_BGR	Transform the color space from CIE XYZ (scaled to the full value range of the image) to BGR.
TCVN_CST_XYZ_TO_RGB	Transform the color space from CIE XYZ (scaled to the full value range of the image) to RGB.
TCVN_CST_BGR_TO_YCRCB	Transform the color space from BGR to YCrCb (scaled to the full value range of the image).
TCVN_CST_RGB_TO_YCRCB	Transform the color space from RGB to YCrCb (scaled to the full value range of the image).
TCVN_CST_YCRCB_TO_BGR	Transform the color space from YCrCb (scaled to the full value range of the image) to BGR.
TCVN_CST_YCRCB_TO_RGB	Transform the color space from YCrCb (scaled to the full value range of the image) to RGB.
TCVN_CST_BGR_TO_HSV	Transform the color space from BGR to HSV (for images of type USINT, H is scaled to a range from 0 to 180 and S,V from 0 to 255).
TCVN_CST_RGB_TO_HSV	Transform the color space from RGB to HSV (for images of type USINT, H is scaled to a range from 0 to 180 and S,V from 0 to 255).
TCVN_CST_BGR_TO_LAB	Transform the color space from BGR to CIE L*a*b* (for images of type USINT, all channels are scaled to a range from 0 to 255).
TCVN_CST_RGB_TO_LAB	Transform the color space from RGB to CIE L*a*b* (for images of type USINT, all channels are scaled to a range from 0 to 255).
TCVN_CST_BAYER_RG_TO_BGR	Transform the color space from BayerRG to BGR.
TCVN_CST_BAYER_BG_TO_RGB	Transform the color space from BayerBG to RGB.
TCVN_CST_BAYER_GR_TO_BGR	Transform the color space from BayerGR to BGR.
TCVN_CST_BAYER_GB_TO_RGB	Transform the color space from BayerGB to RGB.
TCVN_CST_BAYER_BG_TO_BGR	Transform the color space from BayerBG to BGR.
TCVN_CST_BAYER_RG_TO_RGB	Transform the color space from BayerRG to RGB.
TCVN_CST_BAYER_GB_TO_BGR	Transform the color space from BayerGB to BGR.
TCVN_CST_BAYER_GR_TO_RGB	Transform the color space from BayerGR to RGB.
TCVN_CST_BGR_TO_LUV	Transform the color space from BGR to CIE L*u*v* (for images of type USINT, all channels are scaled to a range from 0 to 255).
TCVN_CST_RGB_TO_LUV	Transform the color space from RGB to CIE L*u*v* (for images of type USINT, all channels are scaled to a range from 0 to 255).
TCVN_CST_BGR_TO_HLS	Transform the color space from BGR to HLS (for images of type USINT, H is scaled to a range from 0 to 180 and L,S from 0 to 255).

Name	Description
TCVN_CST_RGB_TO_HLS	Transform the color space from RGB to HLS (for images of type USINT, H is scaled to a range from 0 to 180 and L,S from 0 to 255).
TCVN_CST_HSV_TO_BGR	Transform the color space from HSV (for images of type USINT, H is scaled to a range from 0 to 180 and S,V from 0 to 255) to BGR.
TCVN_CST_HSV_TO_RGB	Transform the color space from HSV (for images of type USINT, H is scaled to a range from 0 to 180 and S,V from 0 to 255) to RGB.
TCVN_CST_LAB_TO_BGR	Transform the color space from CIE L*a*b* (for images of type USINT, all channels are scaled to a range from 0 to 255) to BGR.
TCVN_CST_LAB_TO_RGB	Transform the color space from CIE L*a*b* (for images of type USINT, all channels are scaled to a range from 0 to 255) to RGB.
TCVN_CST_LUV_TO_BGR	Transform the color space from CIE L*u*v* (for images of type USINT, all channels are scaled to a range from 0 to 255) to BGR.
TCVN_CST_LUV_TO_RGB	Transform the color space from CIE L*u*v* (for images of type USINT, all channels are scaled to a range from 0 to 255) to RGB.
TCVN_CST_HLS_TO_BGR	Transform the color space from HLS (for images of type USINT, H is scaled to a range from 0 to 180 and L,S from 0 to 255) to BGR.
TCVN_CST_HLS_TO_RGB	Transform the color space from HLS (for images of type USINT, H is scaled to a range from 0 to 180 and L,S from 0 to 255) to RGB.
TCVN_CST_BAYER_RG_TO_BGR_VNG	Transform the color space from BayerRG to BGR, using Variable Number of Gradients.
TCVN_CST_BAYER_GR_TO_BGR_VNG	Transform the color space from BayerGR to BGR, using Variable Number of Gradients.
TCVN_CST_BAYER_BG_TO_BGR_VNG	Transform the color space from BayerBG to BGR, using Variable Number of Gradients.
TCVN_CST_BAYER_GB_TO_BGR_VNG	Transform the color space from BayerGB to BGR, using Variable Number of Gradients.
TCVN_CST_BAYER_RG_TO_RGB_VNG	Transform the color space from BayerRG to RGB, using Variable Number of Gradients.
TCVN_CST_BAYER_GR_TO_RGB_VNG	Transform the color space from BayerGR to RGB, using Variable Number of Gradients.
TCVN_CST_BAYER_BG_TO_RGB_VNG	Transform the color space from BayerBG to RGB, using Variable Number of Gradients.
TCVN_CST_BAYER_GB_TO_RGB_VNG	Transform the color space from BayerGB to RGB, using Variable Number of Gradients.
TCVN_CST_BGR_TO_HSV_FULL	Transform the color space from BGR to HSV (for images of type USINT, all channels are scaled to a range from 0 to 255).
TCVN_CST_RGB_TO_HSV_FULL	Transform the color space from RGB to HSV (for images of type USINT, all channels are scaled to a range from 0 to 255).
TCVN_CST_BGR_TO_HLS_FULL	Transform the color space from BGR to HLS (for images of type USINT, all channels are scaled to a range from 0 to 255).
TCVN_CST_RGB_TO_HLS_FULL	Transform the color space from RGB to HLS (for images of type USINT, all channels are scaled to a range from 0 to 255).

Name	Description
TCVN_CST_HSV_TO_BGR_FULL	Transform the color space from HSV (for images of type USINT, all channels are scaled to a range from 0 to 255) to BGR.
TCVN_CST_HSV_TO_RGB_FULL	Transform the color space from HSV (for images of type USINT, all channels are scaled to a range from 0 to 255) to RGB.
TCVN_CST_HLS_TO_BGR_FULL	Transform the color space from HLS (for images of type USINT, all channels are scaled to a range from 0 to 255) to BGR.
TCVN_CST_HLS_TO_RGB_FULL	Transform the color space from HLS (for images of type USINT, all channels are scaled to a range from 0 to 255) to RGB.
TCVN_CST_LBGR_TO_LAB	Transform the color space from LBGR to CIE L*a*b* (for images of type USINT, all channels are scaled to a range from 0 to 255).
TCVN_CST_LRGB_TO_LAB	Transform the color space from LRGB to CIE L*a*b* (for images of type USINT, all channels are scaled to a range from 0 to 255).
TCVN_CST_LBGR_TO_LUV	Transform the color space from LBGR to CIE L*u*v* (for images of type USINT, all channels are scaled to a range from 0 to 255).
TCVN_CST_LRGB_TO_LUV	Transform the color space from LRGB to CIE L*u*v* (for images of type USINT, all channels are scaled to a range from 0 to 255).
TCVN_CST_LAB_TO_LBGR	Transform the color space from CIE L*a*b* (for images of type USINT, all channels are scaled to a range from 0 to 255) to LBGR.
TCVN_CST_LAB_TO_LRGB	Transform the color space from CIE L*a*b* (for images of type USINT, all channels are scaled to a range from 0 to 255) to LRGB.
TCVN_CST_LUV_TO_LBGR	Transform the color space from CIE L*u*v* (for images of type USINT, all channels are scaled to a range from 0 to 255) to LBGR.
TCVN_CST_LUV_TO_LRGB	Transform the color space from CIE L*u*v* (for images of type USINT, all channels are scaled to a range from 0 to 255) to LRGB.
TCVN_CST_BGR_TO_YUV	Transform the color space from BGR to YUV (scaled to the full value range of the image).
TCVN_CST_RGB_TO_YUV	Transform the color space from RGB to YUV (scaled to the full value range of the image).
TCVN_CST_YUV_TO_BGR	Transform the color space from YUV (scaled to the full value range of the image) to BGR.
TCVN_CST_YUV_TO_RGB	Transform the color space from YUV (scaled to the full value range of the image) to RGB.
TCVN_CST_BAYER_RG_TO_GRAY	Transform the color space from BayerRG to Gray.
TCVN_CST_BAYER_GR_TO_GRAY	Transform the color space from BayerGR to Gray.
TCVN_CST_BAYER_BG_TO_GRAY	Transform the color space from BayerBG to Gray.
TCVN_CST_BAYER_GB_TO_GRAY	Transform the color space from BayerGB to Gray.
TCVN_CST_YUV_420_NV12_TO_RGB	Transform the color space from YUV420 NV12 to RGB.
TCVN_CST_YUV_420_NV12_TO_BGR	Transform the color space from YUV420 NV12 to BGR.
TCVN_CST_YUV_420_NV21_TO_RGB	Transform the color space from YUV420 NV21 (SP) to RGB.

Name	Description
TCVN_CST_YUV_420_NV21_TO_BGR	Transform the color space from YUV420 NV21 (SP) to BGR.
TCVN_CST_YUV_420_SP_TO_RGB	Transform the color space from YUV420 NV21 (SP) to RGB.
TCVN_CST_YUV_420_SP_TO_BGR	Transform the color space from YUV420 NV21 (SP) to BGR.
TCVN_CST_YUV_420_NV12_TO_RGBA	Transform the color space from YUV420 NV12 to RGBA.
TCVN_CST_YUV_420_NV12_TO_BGRA	Transform the color space from YUV420 NV12 to BGRA.
TCVN_CST_YUV_420_NV21_TO_RGBA	Transform the color space from YUV420 NV21 (SP) to RGBA.
TCVN_CST_YUV_420_NV21_TO_BGRA	Transform the color space from YUV420 NV21 (SP) to BGRA.
TCVN_CST_YUV_420_SP_TO_RGBA	Transform the color space from YUV420 NV21 (SP) to RGBA.
TCVN_CST_YUV_420_SP_TO_BGRA	Transform the color space from YUV420 NV21 (SP) to BGRA.
TCVN_CST_YUV_420_YV12_TO_RGB	Transform the color space from YUV420 YV12 (P) to RGB.
TCVN_CST_YUV_420_YV12_TO_BGR	Transform the color space from YUV420 YV12 (P) to BGR.
TCVN_CST_YUV_420_IYUV_TO_RGB	Transform the color space from YUV420 IYUV (I420) to RGB.
TCVN_CST_YUV_420_IYUV_TO_BGR	Transform the color space from YUV420 IYUV (I420) to BGR.
TCVN_CST_YUV_420_I420_TO_RGB	Transform the color space from YUV420 IYUV (I420) to RGB.
TCVN_CST_YUV_420_I420_TO_BGR	Transform the color space from YUV420 IYUV (I420) to BGR.
TCVN_CST_YUV_420_P_TO_RGB	Transform the color space from YUV420 YV12 (P) to RGB.
TCVN_CST_YUV_420_P_TO_BGR	Transform the color space from YUV420 YV12 (P) to BGR.
TCVN_CST_YUV_420_YV12_TO_RGBA	Transform the color space from YUV420 YV12 (P) to RGBA.
TCVN_CST_YUV_420_YV12_TO_BGRA	Transform the color space from YUV420 YV12 (P) to BGRA.
TCVN_CST_YUV_420_IYUV_TO_RGBA	Transform the color space from YUV420 IYUV (I420) to RGBA.
TCVN_CST_YUV_420_IYUV_TO_BGRA	Transform the color space from YUV420 IYUV (I420) to BGRA.
TCVN_CST_YUV_420_I420_TO_RGBA	Transform the color space from YUV420 IYUV (I420) to RGBA.
TCVN_CST_YUV_420_I420_TO_BGRA	Transform the color space from YUV420 IYUV (I420) to BGRA.
TCVN_CST_YUV_420_P_TO_RGBA	Transform the color space from YUV420 YV12 (P) to RGBA.
TCVN_CST_YUV_420_P_TO_BGRA	Transform the color space from YUV420 YV12 (P) to BGRA.
TCVN_CST_YUV_420_TO_GRAY	Transform the color space from YUV420 to Gray.
TCVN_CST_YUV_420_NV21_TO_GRAY	Transform the color space from YUV420 to Gray.
TCVN_CST_YUV_420_NV12_TO_GRAY	Transform the color space from YUV420 to Gray.
TCVN_CST_YUV_420_YV12_TO_GRAY	Transform the color space from YUV420 to Gray.

Name	Description
TCVN_CST_YUV_420_IYUV_TO_GRAY	Transform the color space from YUV420 to Gray.
TCVN_CST_YUV_420_I420_TO_GRAY	Transform the color space from YUV420 to Gray.
TCVN_CST_YUV_420_SP_TO_GRAY	Transform the color space from YUV420 to Gray.
TCVN_CST_YUV_420_P_TO_GRAY	Transform the color space from YUV420 to Gray.
TCVN_CST_YUV_422_UYVY_TO_RGB	Transform the color space from YUV422 UYVY (Y422, UYNV) to RGB.
TCVN_CST_YUV_422_UYVY_TO_BGR	Transform the color space from YUV422 UYVY (Y422, UYNV) to BGR.
TCVN_CST_YUV_422_Y422_TO_RGB	Transform the color space from YUV422 UYVY (Y422, UYNV) to RGB.
TCVN_CST_YUV_422_Y422_TO_BGR	Transform the color space from YUV422 UYVY (Y422, UYNV) to BGR.
TCVN_CST_YUV_422_UYNV_TO_RGB	Transform the color space from YUV422 UYVY (Y422, UYNV) to RGB.
TCVN_CST_YUV_422_UYNV_TO_BGR	Transform the color space from YUV422 UYVY (Y422, UYNV) to BGR.
TCVN_CST_YUV_422_UYVY_TO_RGBA	Transform the color space from YUV422 UYVY (Y422, UYNV) to RGBA.
TCVN_CST_YUV_422_UYVY_TO_BGRA	Transform the color space from YUV422 UYVY (Y422, UYNV) to BGRA.
TCVN_CST_YUV_422_Y422_TO_RGBA	Transform the color space from YUV422 UYVY (Y422, UYNV) to RGB.
TCVN_CST_YUV_422_Y422_TO_BGRA	Transform the color space from YUV422 UYVY (Y422, UYNV) to BGRA.
TCVN_CST_YUV_422_UYNV_TO_RGBA	Transform the color space from YUV422 UYVY (Y422, UYNV) to RGB.
TCVN_CST_YUV_422_UYNV_TO_BGRA	Transform the color space from YUV422 UYVY (Y422, UYNV) to BGRA.
TCVN_CST_YUV_422_YUY2_TO_RGB	Transform the color space from YUV422 YUYV (YUY2, YUNV) to RGB.
TCVN_CST_YUV_422_YUY2_TO_BGR	Transform the color space from YUV422 YUYV (YUY2, YUNV) to BGR.
TCVN_CST_YUV_422_YVYU_TO_RGB	Transform the color space from YUV422 YVYU to RGB.
TCVN_CST_YUV_422_YVYU_TO_BGR	Transform the color space from YUV422 YVYU to BGR.
TCVN_CST_YUV_422_YUYV_TO_RGB	Transform the color space from YUV422 YUYV (YUY2, YUNV) to RGB.
TCVN_CST_YUV_422_YUYV_TO_BGR	Transform the color space from YUV422 YUYV (YUY2, YUNV) to BGR.
TCVN_CST_YUV_422_YUNV_TO_RGB	Transform the color space from YUV422 YUYV (YUY2, YUNV) to RGB.
TCVN_CST_YUV_422_YUNV_TO_BGR	Transform the color space from YUV422 YUYV (YUY2, YUNV) to BGR.
TCVN_CST_YUV_422_YUY2_TO_RGBA	Transform the color space from YUV422 YUYV (YUY2, YUNV) to RGBA.
TCVN_CST_YUV_422_YUY2_TO_BGRA	Transform the color space from YUV422 YUYV (YUY2, YUNV) to BGRA.
TCVN_CST_YUV_422_YVYU_TO_RGBA	Transform the color space from YUV422 YVYU to RGBA.
TCVN_CST_YUV_422_YVYU_TO_BGRA	Transform the color space from YUV422 YVYU to BGRA.
TCVN_CST_YUV_422_YUYV_TO_RGBA	Transform the color space from YUV422 YUYV (YUY2, YUNV) to RGBA.

Name	Description
TCVN_CST_YUV_422_YUYV_TO_BGRA	Transform the color space from YUV422 YUYV (YUY2, YUNV) to BGRA.
TCVN_CST_YUV_422_YUNV_TO_RGBA	Transform the color space from YUV422 YUYV (YUY2, YUNV) to RGBA.
TCVN_CST_YUV_422_YUNV_TO_BGRA	Transform the color space from YUV422 YUYV (YUY2, YUNV) to BGRA.
TCVN_CST_YUV_422_UYVY_TO_GRAY	Transform the color space from YUV422 UYVY (Y422, UYNV) to Gray.
TCVN_CST_YUV_422_YUY2_TO_GRAY	Transform the color space from YUV422 YUYV (YUY2, YUNV) to Gray.
TCVN_CST_YUV_422_Y422_TO_GRAY	Transform the color space from YUV422 UYVY (Y422, UYNV) to Gray.
TCVN_CST_YUV_422_UYNV_TO_GRAY	Transform the color space from YUV422 UYVY (Y422, UYNV) to Gray.
TCVN_CST_YUV_422_YVYU_TO_GRAY	Transform the color space from YUV422 YVYU to Gray.
TCVN_CST_YUV_422_YUYV_TO_GRAY	Transform the color space from YUV422 YUYV (YUY2, YUNV) to Gray.
TCVN_CST_YUV_422_YUNV_TO_GRAY	Transform the color space from YUV422 YUYV (YUY2, YUNV) to Gray.
TCVN_CST_RGBA_TO_PREMULTIPLICATED_RGBA	Transform the color space from RGBA to premultiplicated RGBA.
TCVN_CST_PREMULTIPLICATED_RGBA_TO_RGBA	Transform the color space from premultiplicated RGBA to RGBA.
TCVN_CST_RGB_TO_YUV_420_I420	Transform the color space from RGB to YUV420 IYUV (I420).
TCVN_CST_BGR_TO_YUV_420_I420	Transform the color space from BGR to YUV420 IYUV (I420).
TCVN_CST_RGB_TO_YUV_420_IYUV	Transform the color space from RGB to YUV420 IYUV (I420).
TCVN_CST_BGR_TO_YUV_420_IYUV	Transform the color space from BGR to YUV420 IYUV (I420).
TCVN_CST_RGBA_TO_YUV_420_I420	Transform the color space from RGBA to YUV420 IYUV (I420).
TCVN_CST_BGRA_TO_YUV_420_I420	Transform the color space from BGRA to YUV420 IYUV (I420).
TCVN_CST_RGBA_TO_YUV_420_IYUV	Transform the color space from RGBA to YUV420 IYUV (I420).
TCVN_CST_BGRA_TO_YUV_420_IYUV	Transform the color space from BGRA to YUV420 IYUV (I420).
TCVN_CST_RGB_TO_YUV_420_YV12	Transform the color space from RGB to YUV420 YV12 (P).
TCVN_CST_BGR_TO_YUV_420_YV12	Transform the color space from BGR to YUV420 YV12 (P).
TCVN_CST_RGBA_TO_YUV_420_YV12	Transform the color space from RGBA to YUV420 YV12 (P).
TCVN_CST_BGRA_TO_YUV_420_YV12	Transform the color space from BGRA to YUV420 YV12 (P).
TCVN_CST_BAYER_RG_TO_BGR_EA	Transform the color space from BayerRG to BGR, using an Edge Aware algorithm.
TCVN_CST_BAYER_GR_TO_BGR_EA	Transform the color space from BayerGR to BGR, using an Edge Aware algorithm.
TCVN_CST_BAYER_BG_TO_BGR_EA	Transform the color space from BayerBG to BGR, using an Edge Aware algorithm.

Name	Description
TCVN_CST_BAYER_GB_TO_BGR_EA	Transform the color space from BayerGB to BGR, using an Edge Aware algorithm.
TCVN_CST_BAYER_RG_TO_RGB_EA	Transform the color space from BayerRG to RGB, using an Edge Aware algorithm.
TCVN_CST_BAYER_GR_TO_RGB_EA	Transform the color space from BayerGR to RGB, using an Edge Aware algorithm.
TCVN_CST_BAYER_BG_TO_RGB_EA	Transform the color space from BayerBG to RGB, using an Edge Aware algorithm.
TCVN_CST_BAYER_GB_TO_RGB_EA	Transform the color space from BayerGB to RGB, using an Edge Aware algorithm.
TCVN_CST_BAYER_RG_TO_BGRA	Transform the color space from BayerRG to BGRA.
TCVN_CST_BAYER_GR_TO_BGRA	Transform the color space from BayerGR to BGRA.
TCVN_CST_BAYER_BG_TO_BGRA	Transform the color space from BayerBG to BGRA.
TCVN_CST_BAYER_GB_TO_BGRA	Transform the color space from BayerGB to BGRA.
TCVN_CST_BAYER_RG_TO_RGBA	Transform the color space from BayerRG to RGBA.
TCVN_CST_BAYER_GR_TO_RGBA	Transform the color space from BayerGR to RGBA.
TCVN_CST_BAYER_BG_TO_RGBA	Transform the color space from BayerBG to RGBA.
TCVN_CST_BAYER_GB_TO_RGBA	Transform the color space from BayerGB to RGBA.
TCVN_CST_MAX	For internal use only, adapted when adding new values.

Verwandte Funktionen

- F_VN_ConvertColorSpace [▶ 1270]

6.1.2.4.15 ETcVnColorTrainingMethod

Offers color training methods.

Syntax

Definition:

```
TYPE ETcVnColorTrainingMethod :
(
    TCVN_CTM_LAB := 0,
    TCVN_CTM_RGB := 1
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_CTM_LAB	Trains the color in CIE L*a*b* color space.
TCVN_CTM_RGB	Trains the color in RGB color space.

Verwandte Funktionen

- F_VN_TrainImageColorExp [▶ 1293]

6.1.2.4.16 ETcVnConnectedComponentsAlgorithm

Offers connected components algorithms.

Syntax

Definition:

```
TYPE ETcVnConnectedComponentsAlgorithm :
(
    TCVN_CCA_WU    := 0,
    TCVN_CCA_GRANA := 1
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_CCA_WU	SAUF algorithm
TCVN_CCA_GRANA	BBDT algorithm for 8-way connectivity, SAUF algorithm for 4-way connectivity

6.1.2.4.17 ETcVnContainerExportFormat

Offers container export formats.

Syntax

Definition:

```
TYPE ETcVnContainerExportFormat :
(
    TCVN_CEF_XML            := 0,
    TCVN_CEF_XML_SERIALIZED := 1,
    TCVN_CEF_CSV            := 2,
    TCVN_CEF_BINARY         := 3
)UINT;
END_TYPE
```

Values

Name	Description
TCVN_CEF_XML	XML format with human readable data
TCVN_CEF_XML_SERIALIZED	XML format with serialized data
TCVN_CEF_CSV	CSV format (limited to two dimensional representation, so not applicable for all container types)
TCVN_CEF_BINARY	Binary serialized data (recommended for large containers)

Verwandte Elemente

- FB_VN_WriteContainer [▶ 1654]

6.1.2.4.18 ETcVnContourApproximationMethod

Offers methods for contour approximation.

Syntax

Definition:

```
TYPE ETcVnContourApproximationMethod :
(
    TCVN_CAM_NONE      := 1,
    TCVN_CAM_SIMPLE    := 2,
    TCVN_CAM_TC89_L1   := 3,
    TCVN_CAM_TC89_KCOS := 4
)DINT;
END_TYPE
```

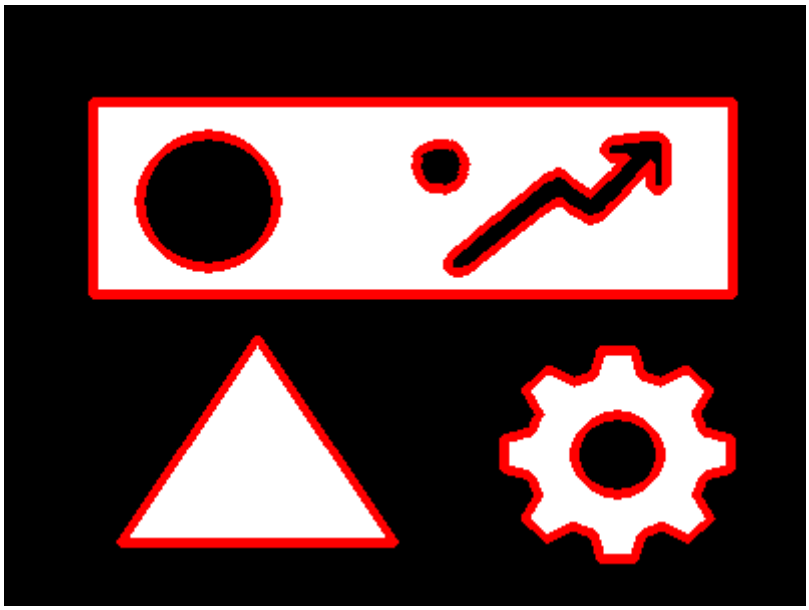
Values

Name	Description
TCVN_CAM_NONE	No approximation, every single point is stored, i.e. two subsequent points are always direct neighbors (horizontal, vertical or diagonal).
TCVN_CAM_SIMPLE	Lossless compression of segments that form straight lines in horizontal, vertical or diagonal direction leaving only their endpoints (For instance, an up-right rectangle is reduced to its four corner points.)
TCVN_CAM_TC89_L1	Compression using a Teh-Chin chain approximation algorithm (IEEE-paper C. Teh, R. Chin, On the Detection of Dominant Points on Digital Curves, 1989).
TCVN_CAM_TC89_KCOS	Compression using a Teh-Chin chain approximation algorithm (IEEE-paper C. Teh, R. Chin, On the Detection of Dominant Points on Digital Curves, 1989).

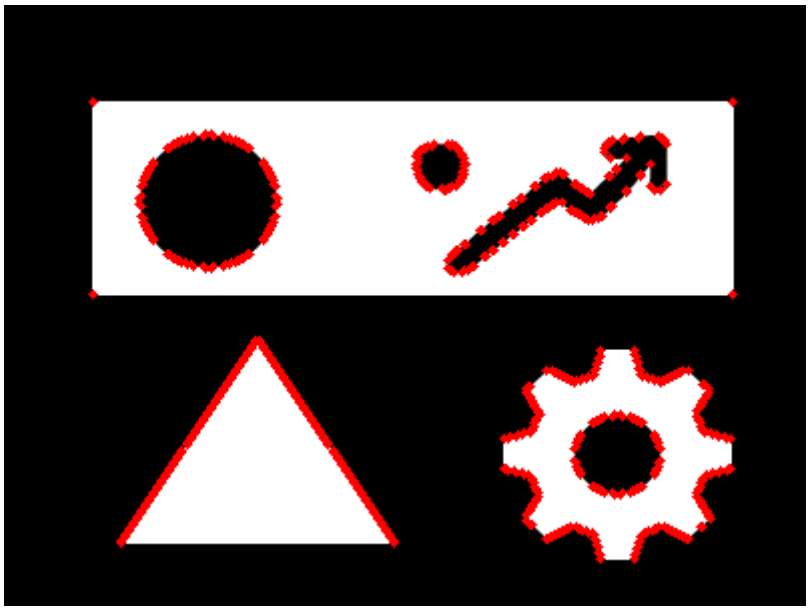
Weiterführende Informationen

Über diesen Typ wird angegeben, ob - und wenn ja - wie die eine Kontur beschreibenden Punkte vereinfacht werden sollen.

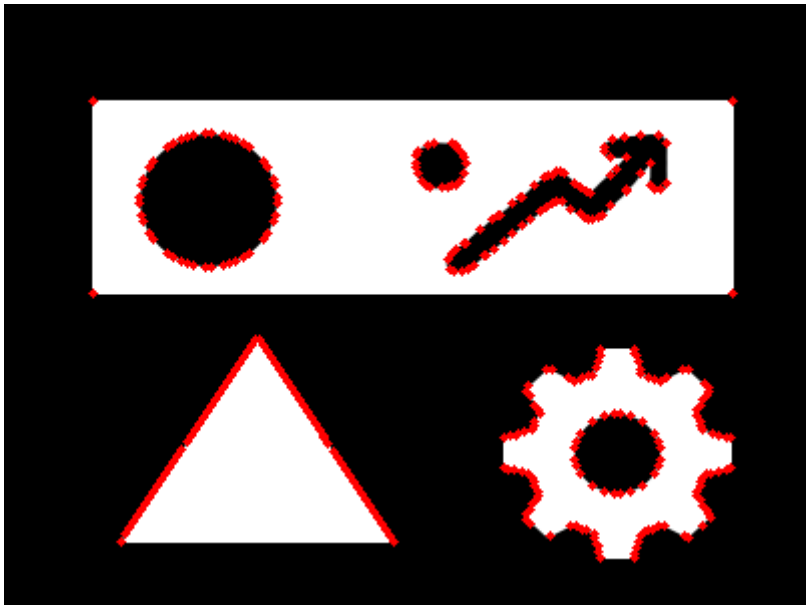
- **TCVN_CAM_NONE**
  Keine Vereinfachung der Konturbeschreibung, die Konturen werden anhand aller umschließender Punkte beschrieben.



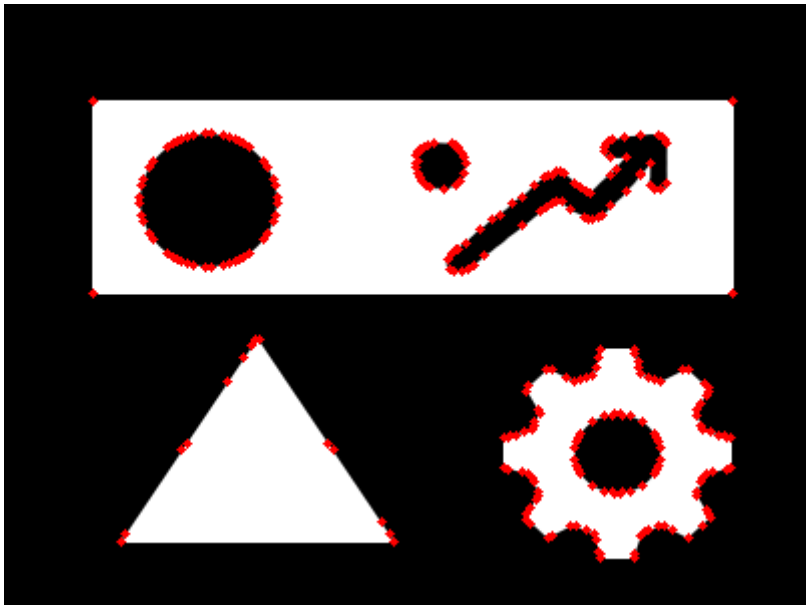
- **TCVN_CAM_SIMPLE**
  Verlustfreie Vereinfachung



- **TCVN_CAM_TC89_L1**
  Vereinfachung der Konturpunkte mithilfe des Teh-Chin chain Approximation Algorithmus (IEEE-paper C. Teh, R. Chin, On the Detection of Dominant Points on Digital Curves, 1989).



- **TCVN_CAM_TC89_KCOS**
  Vereinfachung der Konturpunkte mithilfe des Teh-Chin chain Approximation Algorithmus (IEEE-paper C. Teh, R. Chin, On the Detection of Dominant Points on Digital Curves, 1989).



Verwandte Funktionen

- F_VN_FindContoursExp [▶ 1219]
- F_VN_FindContourHierarchyExp [▶ 1212]

6.1.2.4.19 ETcVnContourRetrievalMode

Offers retrieval modes for a contour search.

Syntax

Definition:

```
TYPE ETcVnContourRetrievalMode :
(
    TCVN_CRM_EXTERNAL             := 0,
    TCVN_CRM_LIST                 := 1,
    TCVN_CRM_CONNECTED_COMPONENTS := 2,
    TCVN_CRM_TREE                 := 3,
    TCVN_CRM_FLOODFILL            := 4
)DINT;
END_TYPE
```

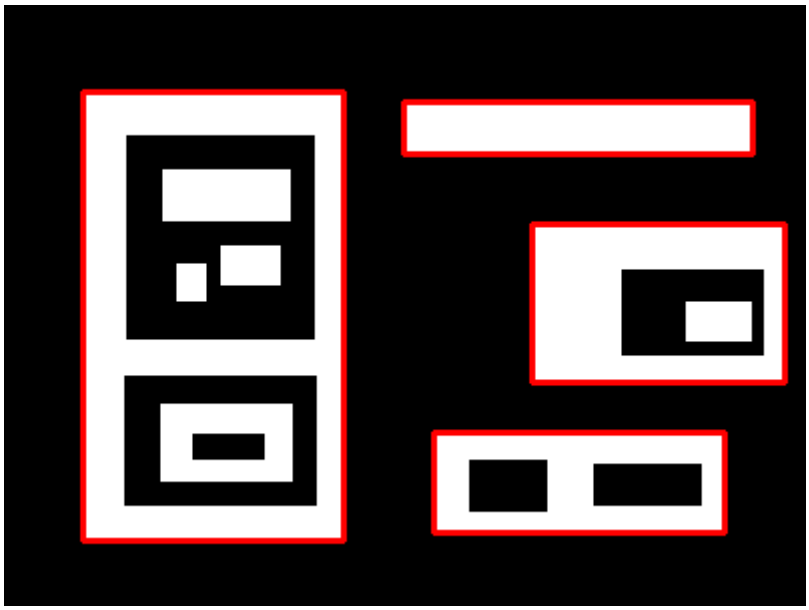
Values

Name	Description
TCVN_CRM_EXTERNAL	Returns only external contours.
TCVN_CRM_LIST	Returns all found contours, not considering their hierarchy.
TCVN_CRM_CONNECTED_COMPONENTS	Returns all contours with a 2-level hierarchy (External contours are assigned level 0, internal contours level 1).
TCVN_CRM_TREE	Returns all contours and their full hierarchy.
TCVN_CRM_FLOODFILL	Returns the found contours using a floodfill algorithm (only available for DINT images).

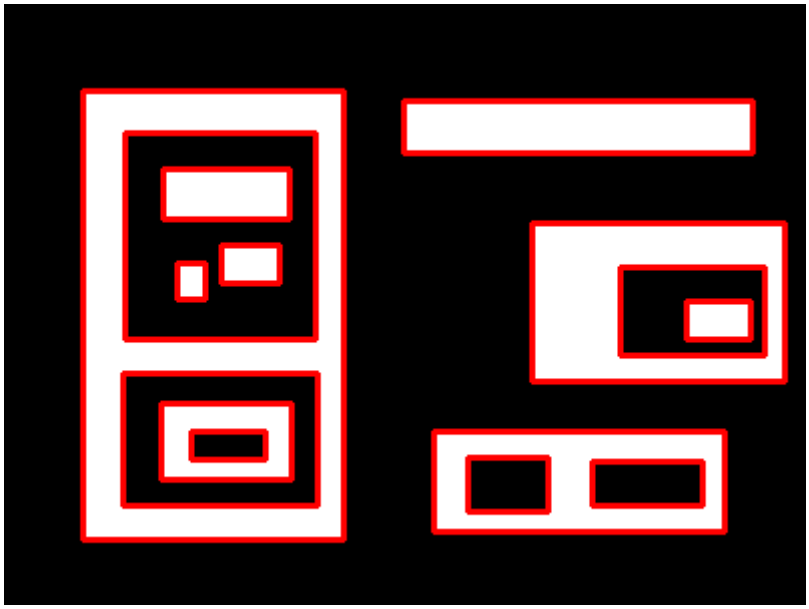
Weiterführende Informationen

Über diesen Typ wird angegeben wie die Kontur-Hierarchie bei der Kontursuche berücksichtigt werden soll.

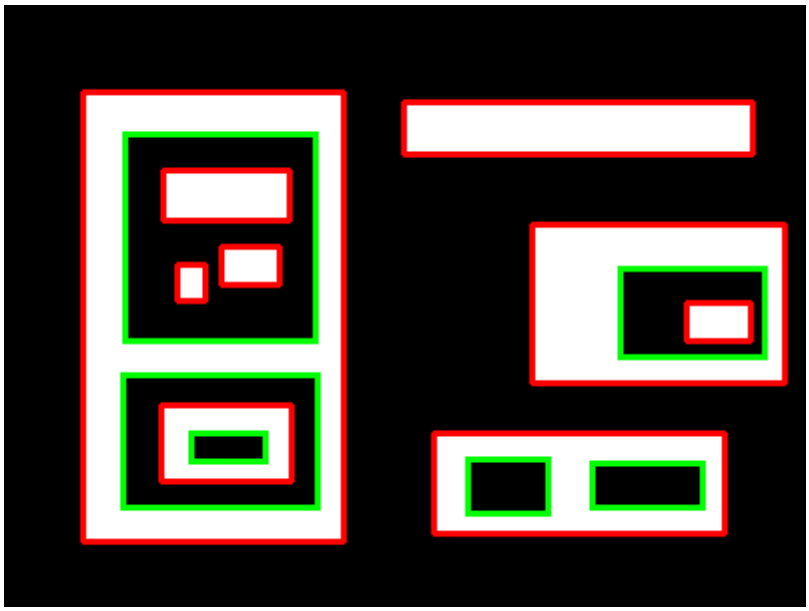
- **TCVN_CRM_EXTERNAL**
  Nur die äußeren Konturen werden zurückgegeben, innere Konturen werden nicht berücksichtigt.



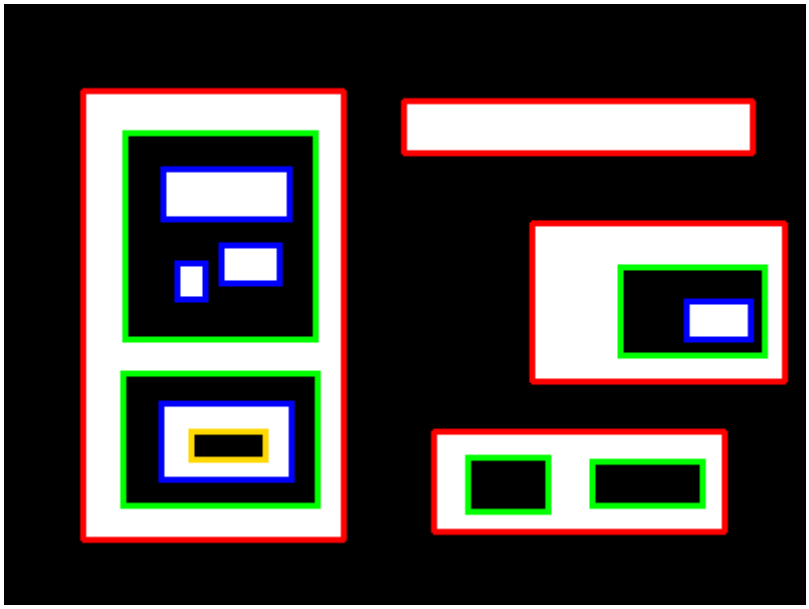
- **TCVN_CRM_LIST**
  Alle Konturen werden zurückgegeben. Die Hierarchie der Konturen wird dabei nicht berücksichtigt.



- **TCVN_CRM_CONNECTED_COMPONENTS**
  Alle Konturen werden zurückgegeben und dazu eine 2-Level-Hierarchy erstellt. In der Grafik sind alle Konturen auf Ebene 1 (Level 1) rot gezeichnet und auf Ebene 2 (Level 2) grün. So ist gut zu erkennen, dass nach einer Kontur auf Ebene 2 die nächste innenliegende Kontur wieder auf Ebene 1 liegt.



- **TCVN_CRM_TREE**
  Alle Konturen werden zurückgegeben und dazu eine Hierarchie entsprechend einer Baumstruktur erstellt. Im Bild Ebene 1 rot, Ebene 2 grün, Ebene 3 blau und Ebene 4 gelb.



Verwandte Funktionen

- F_VN_FindContoursExp [▶ 1219]
- F_VN_FindContourHierarchyExp [▶ 1212]

6.1.2.4.20 ETcVnContoursMatchComparisonMethod

Offers comparison methods for contour matching.

Syntax

Definition:

```
TYPE ETcVnContoursMatchComparisonMethod :
(
    TCVN_CMCM_CONTOURS_MATCH_I1 := 1,
    TCVN_CMCM_CONTOURS_MATCH_I2 := 2,
    TCVN_CMCM_CONTOURS_MATCH_I3 := 3
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_CMCM_CONTOURS_MATCH_I1	Contours are compared using the sum over the differences between the reciprocal individual characteristics.
TCVN_CMCM_CONTOURS_MATCH_I2	Contours are compared using the sum over the differences between the individual characteristics.
TCVN_CMCM_CONTOURS_MATCH_I3	Contours are compared using only the maximum difference between the individual characteristics.

Weiterführende Informationen

Die verfügbaren Methoden basieren auf Hu-Invarianten. Sie verwenden die folgenden Algorithmen, wobei A für die erste und B für die zweite Kontur steht.

$$I_1(A,B) = \sum_{i=1..7} \left| \frac{1}{m_i^A} - \frac{1}{m_i^B} \right|$$

$$I_2(A,B) = \sum_{i=1..7} \left| m_i^A - m_i^B \right|$$

$$I_3(A,B) = \max_{i=1..7} \frac{\left| m_i^A - m_i^B \right|}{\left| m_i^A \right|}$$

$$m_i^A = sign(h_i^A) \cdot \log(h_i^A)$$

$$m_i^B = sign(h_i^B) \cdot \log(h_i^B)$$

h_i^A und h_i^B mit `i` in `[1, 7]` sind jeweils Hu Momente von A und B.

Verwandte Funktionen

- F_VN_MatchContours(Exp) [▶ 1013]
- F_VN_MatchContours1vsN(Exp) [▶ 1014]
- F_VN_MatchImageHuMoments [▶ 1227]

6.1.2.4.21 ETcVnDiffusivityTypeKAZE

Offers diffusivity types for feature detection methods KAZE and AKAZE.

Syntax

Definition:

```
TYPE ETcVnDiffusivityTypeKAZE :
(
    TCVN_DT1_KAZE_PM_G1        := 0,
    TCVN_DT1_KAZE_PM_G2        := 1,
    TCVN_DT1_KAZE_WEICKERT     := 2,
    TCVN_DT1_KAZE_CHARBONNIER := 3
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_DT1_KAZE_PM_G1	Uses the Perona-Malik diffusivity filter g1.
TCVN_DT1_KAZE_PM_G2	Uses the Perona-Malik diffusivity filter g2.
TCVN_DT1_KAZE_WEICKERT	Uses the Weickert diffusivity filter.
TCVN_DT1_KAZE_CHARBONNIER	Uses the Charbonnier diffusivity filter.

Weiterführende Informationen

Das Enum `ETcVnDiffusivityTypeKAZE` wird in der Struktur TcVnParamsKAZE [▶ 253] verwendet.

Verwandte Funktionen

- F_VN_KeyPointsAndDescriptorsKAZEExp [▶ 1392]

6.1.2.4.22 ETcVnDistanceTransformationLabel

Offers types of the label array to build.

Syntax

Definition:

```
TYPE ETcVnDistanceTransformationLabel :
(
    TCVN_DTL_CCOMP := 0,
    TCVN_DTL_PIXEL := 1
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_DTL_CCOMP	Labels connected components (Each connected component of zeros in the source image and the pixels closest to the connected component will be assigned the same label).
TCVN_DTL_PIXEL	Labels pixels (Each zero pixel and the non-zero pixels closest to it get their own label).

Verwandte Funktionen

- F_VN_DistanceTransformationExp [▶ 1244]

6.1.2.4.23 ETcVnDistanceTransformationMask

Offers different sizes of the distance transformation mask. Please note, that some sizes are not supported by certain distance types.

Syntax

Definition:

```
TYPE ETcVnDistanceTransformationMask :
(
    TCVN_DTM_PRECISE := 0,
    TCVN_DTM_3       := 3,
    TCVN_DTM_5       := 5
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_DTM_PRECISE	Uses a precise mask.
TCVN_DTM_3	Uses a 3x3 mask.
TCVN_DTM_5	Uses a 5x5 mask.

Verwandte Funktionen

- F_VN_DistanceTransformation(Exp) [▶ 1242]

6.1.2.4.24 ETcVnDistanceType

Offers distance types.

Syntax

Definition:

```
TYPE ETcVnDistanceType :
(
    TCVN_DT_USER   := -1,
    TCVN_DT_L1     := 1,
    TCVN_DT_L2     := 2,
    TCVN_DT_C      := 3,
    TCVN_DT_L12    := 4,
    TCVN_DT_FAIR   := 5,
    TCVN_DT_WELSCH := 6,
    TCVN_DT_HUBER  := 7
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_DT_USER	User defined distance
TCVN_DT_L1	|x1 - x2| + |y1 - y2|
TCVN_DT_L2	Euclidean distance
TCVN_DT_C	max(|x1 - x2|, |y1 - y2|)
TCVN_DT_L12	L1 - L2 : 2*(sqrt(1 + x*x/2) - 1))
TCVN_DT_FAIR	c^2(|x|/c-log(1+|x|/c)), c = 1.3998
TCVN_DT_WELSCH	c^2/2(1-exp(-(x/c)^2)), c = 2.9846
TCVN_DT_HUBER	|x|<c ? x^2/2 : c(|x|-c/2), c=1.345

Verwandte Funktionen

- F_VN_DistanceTransformation(Exp) [▶ 1242]
- F_VN_FitLineExp [▶ 1011]

6.1.2.4.25 ETcVnDrawMatchesFlags

Offers a combination of flags to support overdrawing an existing destination image and/or skipping single (i.e. non-matched) keypoints and/or drawing additional (rich-)keypoint information (size and orientation). Used by F_VN_DrawKeypointsExp.

Syntax

Definition:

```
TYPE ETcVnDrawMatchesFlags :
(
    TCVN_DMF_DEFAULT                          := 0,
    TCVN_DMF_OVERDRAW                         := 1,
    TCVN_DMF_SKIPSINGLE                       := 2,
    TCVN_DMF_OVERDRAW_SKIPSINGLE              := 3,
    TCVN_DMF_RICHKEYPOINT                     := 4,
    TCVN_DMF_OVERDRAW_RICHKEYPOINT            := 5,
    TCVN_DMF_SKIPSINGLE_RICHKEYPOINT          := 6,
    TCVN_DMF_OVERDRAW_SKIPSINGLE_RICHKEYPOINT := 7
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_DMF_DEFAULT	Draw all keypoints into a new image.
TCVN_DMF_OVERDRAW	Draw all keypoints into the existing destination image.
TCVN_DMF_SKIPSINGLE	Draw the keypoints into a new image but skip single keypoints.
TCVN_DMF_OVERDRAW_SKIPSINGLE	Draw the keypoints into the existing destination image but skip single keypoints.
TCVN_DMF_RICHKEYPOINT	Draw all keypoints with rich information into a new image.
TCVN_DMF_OVERDRAW_RICHKEYPOINT	Draw all keypoints with rich information into the existing destination image.
TCVN_DMF_SKIPSINGLE_RICHKEYPOINT	Draw the keypoints with rich information into a new image but skip single keypoints.
TCVN_DMF_OVERDRAW_SKIPSINGLE_RICHKEYPOINT	Draw the keypoints with rich information into the existing destination image but skip single keypoints.

Verwandte Funktionen

- F_VN_DrawKeypointsExp [▶ 1046]
- F_VN_DrawMatchesExp [▶ 1058]

6.1.2.4.26 ETcVnDrawShape

Offers shapes to be drawn.

Syntax

Definition:

```
TYPE ETcVnDrawShape :
(
    TCVN_DS_RANDOM  := -1,
    TCVN_DS_CIRCLE  := 0,
    TCVN_DS_SQUARE  := 1,
    TCVN_DS_PLUS    := 2,
    TCVN_DS_X       := 3,
    TCVN_DS_DIAMOND := 4,
    TCVN_DS_MAX     := 5
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_DS_RANDOM	Randomly select a circle, square, plus, X or diamond.
TCVN_DS_CIRCLE	Circle
TCVN_DS_SQUARE	Square
TCVN_DS_PLUS	+
TCVN_DS_X	x
TCVN_DS_DIAMOND	Diamond
TCVN_DS_MAX	For internal use only, adapted when adding new values.

Verwandte Funktionen

- F_VN_DrawPoint(Exp) [▶ 1062]
- F_VN_DrawPoints(Exp) [▶ 1064]

6.1.2.4.27 ETcVnEdgeDetectionAlgorithm

Offers edge detection algorithms.

Syntax

Definition:

```
TYPE ETcVnEdgeDetectionAlgorithm :
(
    TCVN_EDA_INTERPOLATION   := 0,
    TCVN_EDA_APPROX_ERF      := 1,
    TCVN_EDA_APPROX_GAUSSIAN := 2
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_EDA_INTERPOLATION	Interpolates pixels (bilinear) and then finds the maximum gradient. This approach is fast and stable, but usually less precise than the function approximation methods.
TCVN_EDA_APPROX_ERF	Approximates the edge with an erf function. This approach is slower than the interpolation method, but usually more precise. However, it can be inaccurate if the edge does not suit the erf model.
TCVN_EDA_APPROX_GAUSSIAN	Approximates the edge with a gaussian function. This method is intended to find the center of relatively thin lines, so it is likely to be inaccurate for other edges.

Verwandte Funktionen

- Measurement [▶ 1511]
  - F_VN_LocateCircularArcExp [▶ 1523]
  - F_VN_LocateEdgeExp [▶ 1529]
  - F_VN_LocateEdgesExp [▶ 1535]
  - F_VN_LocateEllipseExp [▶ 1541]
  - F_VN_MeasureAngleBetweenEdgesExp [▶ 1546]
  - F_VN_MeasureEdgeDistanceExp [▶ 1551]
  - F_VN_MeasureMinEdgeDistanceExp [▶ 1556]

6.1.2.4.28 ETcVnEdgeDirection

Offers edge directions relative to the search direction.

Syntax

Definition:

```
TYPE ETcVnEdgeDirection :
(
    TCVN_ED_DARK_TO_LIGHT := 0,
    TCVN_ED_LIGHT_TO_DARK := 1
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_ED_DARK_TO_LIGHT	Dark to light
TCVN_ED_LIGHT_TO_DARK	Light to dark

Verwandte Funktionen

- Measurement [▶ 1511]

6.1.2.4.29 ETcVnElementType

Offers element types.

Syntax

Definition:

```
TYPE ETcVnElementType :
(
    TCVN_ET_SAME_AS_SOURCE := -1,
    TCVN_ET_USINT          := 0,
    TCVN_ET_SINT           := 1,
    TCVN_ET_UINT           := 2,
    TCVN_ET_INT            := 3,
    TCVN_ET_DINT           := 4,
    TCVN_ET_REAL           := 5,
    TCVN_ET_LREAL          := 6
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_ET_SAME_AS_SOURCE	Sets the element type of the destination image to the source image element type.
TCVN_ET_USINT	USINT (depth: 8 bit, 0..255)
TCVN_ET_SINT	SINT (depth: 8 bit, -128..127)
TCVN_ET_UINT	UINT (depth: 16 bit, 0..65535)
TCVN_ET_INT	INT (depth: 16 bit, -32768..32767)
TCVN_ET_DINT	DINT (depth: 32 bit, -2147483648..2147483647)
TCVN_ET_REAL	REAL (depth: 32 bit, ~ -3.402823E 10^38 .. ~ 3.402823E 38)
TCVN_ET_LREAL	LREAL (depth: 64 bit, ~ -1.79769313486231E 308 .. ~ 1.79769313486232E 308)

Weiterführende Informationen

Das Enum `ETcVnElementType` wird hauptsächlich zur Angabe des Pixeltyps bzw. der Bittiefe von Bildern verwendet. Des Weiteren ist es noch in der Struktur TcVnMatrix [▶ 245] enthalten.

Verwandte Funktionen

- F_VN_ConvertElementType(Exp) [▶ 798]
- F_VN_CreateImage [▶ 806]
- F_VN_CreateImageAndSetPixels [▶ 807]
- F_VN_CreateImageFromArray [▶ 808]
- F_VN_CustomFilter(Exp) [▶ 1316]
- F_VN_InitMatrixStruct [▶ 1570]
- F_VN_LaplacianFilter(Exp) [▶ 1328]
- F_VN_NormalizeImageExp [▶ 1281]
- F_VN_ScharrFilter(Exp) [▶ 1342]
- F_VN_SeparableCustomFilter(Exp) [▶ 1346]
- F_VN_SobelFilter(Exp) [▶ 1350]

6.1.2.4.30 ETcVnEstimationAlgorithm

Offers estimation algorithms for matching point sets.

Syntax

Definition:

```
TYPE ETcVnEstimationAlgorithm :
(
    TCVN_EA_DEFAULT := 0,
    TCVN_EA_LMEDS   := 4,
    TCVN_EA_RANSAC  := 8,
    TCVN_EA_RHO     := 16
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_EA_DEFAULT	Use all points.
TCVN_EA_LMEDS	Least Median of Squares
TCVN_EA_RANSAC	Random Sample Consensus
TCVN_EA_RHO	Progressive Sample Consensus

Verwandte Funktionen

- F_VN_FindReferenceKeyPointsInImageORBExp [▶ 1381]
- F_VN_HomographyExp [▶ 1177]

6.1.2.4.31 ETcVnExtremePointDirection

Offers search directions for the extreme point.

Syntax

Definition:

```
TYPE ETcVnExtremePointDirection :
(
    TCVN_EPD_TOP_LEFT      := 0,
    TCVN_EPD_TOP_MEDIAN    := 1,
    TCVN_EPD_TOP_RIGHT     := 2,
    TCVN_EPD_BOTTOM_LEFT   := 3,
    TCVN_EPD_BOTTOM_MEDIAN := 4,
    TCVN_EPD_BOTTOM_RIGHT  := 5,
    TCVN_EPD_LEFT_TOP      := 6,
    TCVN_EPD_LEFT_MEDIAN   := 7,
    TCVN_EPD_LEFT_BOTTOM   := 8,
    TCVN_EPD_RIGHT_TOP     := 9,
    TCVN_EPD_RIGHT_MEDIAN  := 10,
    TCVN_EPD_RIGHT_BOTTOM  := 11
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_EPD_TOP_LEFT	Find the topmost point (min y, take left one if more than 1).
TCVN_EPD_TOP_MEDIAN	Find the topmost point (min y, take median one if more than 1).
TCVN_EPD_TOP_RIGHT	Find the topmost point (min y, take right one if more than 1).
TCVN_EPD_BOTTOM_LEFT	Find the bottommost point (max y, take left one if more than 1).
TCVN_EPD_BOTTOM_MEDIAN	Find the bottommost point (max y, take median one if more than 1).
TCVN_EPD_BOTTOM_RIGHT	Find the bottommost point (max y, take right one if more than 1).
TCVN_EPD_LEFT_TOP	Find the leftmost point (min x, take top one if more than 1).
TCVN_EPD_LEFT_MEDIAN	Find the leftmost point (min x, take median one if more than 1).
TCVN_EPD_LEFT_BOTTOM	Find the leftmost point (min x, take bottom one if more than 1).
TCVN_EPD_RIGHT_TOP	Find the rightmost point (max x, take top one if more than 1).
TCVN_EPD_RIGHT_MEDIAN	Find the rightmost point (max x, take median one if more than 1).
TCVN_EPD_RIGHT_BOTTOM	Find the rightmost point (max x, take bottom one if more than 1).

Verwandte Funktionen

- F_VN_ContourExtremePoint [▶ 994]

6.1.2.4.32 ETcVnFeatureDescriptorTypeAKAZE

Offers descriptor types for AKAZE method.

Syntax

Definition:

```
TYPE ETcVnFeatureDescriptorTypeAKAZE :
(
    TCVN_FDT_AKAZE_KAZE_UPRIGHT := 2,
    TCVN_FDT_AKAZE_KAZE         := 3,
    TCVN_FDT_AKAZE_MLDB_UPRIGHT := 4,
    TCVN_FDT_AKAZE_MLDB         := 5
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_FDT_AKAZE_KAZE_UPRIGHT	KAZE descriptor (upright, i.e. rotation dependend).
TCVN_FDT_AKAZE_KAZE	KAZE descriptor (rotation invariant).
TCVN_FDT_AKAZE_MLDB_UPRIGHT	MLDB descriptor (upright, i.e. rotation dependend).
TCVN_FDT_AKAZE_MLDB	MLDB descriptor (rotation invariant).

Weiterführende Informationen

Das Enum `ETcVnFeatureDescriptorTypAKAZE` wird in der Struktur TcVnParamsAKAZE [▶ 247] verwendet.

6.1.2.4.33 ETcVnFeatureScalingType

Offers feature scaling types

Syntax

Definition:

```
TYPE ETcVnFeatureScalingType :
(
    TCVN_FST1_MAXABS          := 0,
    TCVN_FST1_MINMAX          := 1,
    TCVN_FST1_STANDARDIZATION := 2
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_FST1_MAXABS	Use the max of the absolute values of the feature vector. The normlized values are in the range [-1 to 1] or [0 to 1] if all the original values are positive.
TCVN_FST1_MINMAX	Use the min and max values of the feature vector. The normlized values are in the range [0 to 1]
TCVN_FST1_STANDARDIZATION	Use the mean and the standard deviation of the feature vector. The normalized vector has a mean value equal to zero and standard deviation equal to one

6.1.2.4.34 ETcVnFeatureScoreTypeORB

Offers algorithms used to rank features.

Syntax

Definition:

```
TYPE ETcVnFeatureScoreTypeORB :
(
    TCVN_FST_ORB_HARRIS := 0,
    TCVN_FST_ORB_FAST   := 1
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_FST_ORB_HARRIS	Harris algorithm (produces more stable keypoints, but computation takes longer).
TCVN_FST_ORB_FAST	FAST algorithm (produces slightly less stable keypoints, but computation is faster).

Weiterführende Informationen

Das Enum `ETcVnFeatureScoreTypeORB` wird in der Struktur TcVnParamsORB [▶ 254] verwendet.

6.1.2.4.35 ETcVnFilterDirection

Offers directions, in which to apply filter.

Syntax

Definition:

```
TYPE ETcVnFilterDirection :
(
    TCVN_FD_X := 0,
    TCVN_FD_Y := 1
)INT;
END_TYPE
```

Values

Name	Description
TCVN_FD_X	X-direction
TCVN_FD_Y	Y-direction

Verwandte Funktionen

- F_VN_ScharrFilter(Exp) [▶ 1342]

6.1.2.4.36 ETcVnFlipAxis

Defines the axis around which to flip (mirror) the image.

Syntax

Definition:

```
TYPE ETcVnFlipAxis :
(
    TCVN_FA_XY := -1,
    TCVN_FA_X  := 0,
    TCVN_FA_Y  := 1
)INT;
END_TYPE
```

Values

Name	Description
TCVN_FA_XY	Flip the image around both axes
TCVN_FA_X	Flip the image around the X axis
TCVN_FA_Y	Flip the image around the Y axis

Verwandte Funktionen

- F_VN_FlipImage [▶ 1170]

6.1.2.4.37 ETcVnFontType

Offers font types.

Syntax

Definition:

```
TYPE ETcVnFontType :
(
    TCVN_FT_HERSHEY_SIMPLEX              := 0,
    TCVN_FT_HERSHEY_PLAIN                := 1,
    TCVN_FT_HERSHEY_DUPLEX               := 2,
    TCVN_FT_HERSHEY_COMPLEX              := 3,
    TCVN_FT_HERSHEY_TRIPLEX              := 4,
    TCVN_FT_HERSHEY_COMPLEX_SMALL        := 5,
    TCVN_FT_HERSHEY_SCRIPT_SIMPLEX       := 6,
    TCVN_FT_HERSHEY_SCRIPT_COMPLEX       := 7,
    TCVN_FT_HERSHEY_PLAIN_ITALIC         := 17,
    TCVN_FT_HERSHEY_COMPLEX_ITALIC       := 19,
    TCVN_FT_HERSHEY_TRIPLEX_ITALIC       := 20,
    TCVN_FT_HERSHEY_COMPLEX_SMALL_ITALIC := 21
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_FT_HERSHEY_SIMPLEX	Normal size sans-serif font.
TCVN_FT_HERSHEY_PLAIN	Small size sans-serif font.
TCVN_FT_HERSHEY_DUPLEX	More complex normal size sans-serif font.
TCVN_FT_HERSHEY_COMPLEX	Normal size serif font.
TCVN_FT_HERSHEY_TRIPLEX	More complex normal size serif font.
TCVN_FT_HERSHEY_COMPLEX_SMALL	Small size serif font.
TCVN_FT_HERSHEY_SCRIPT_SIMPLEX	Hand-writing style font.
TCVN_FT_HERSHEY_SCRIPT_COMPLEX	More complex hand-writing style font.
TCVN_FT_HERSHEY_PLAIN_ITALIC	Small size sans-serif font (italic).
TCVN_FT_HERSHEY_COMPLEX_ITALIC	Normal size serif font (italic).
TCVN_FT_HERSHEY_TRIPLEX_ITALIC	More complex normal size serif font (italic).
TCVN_FT_HERSHEY_COMPLEX_SMALL_ITALIC	Small size serif font (italic).

Weiterführende Informationen

Folgende Schriftarten stehen zur Verfügung:

Verwandte Funktionen

- F_VN_PutLabelExp [▶ 1083]
- F_VN_PutText(Exp) [▶ 1084]

6.1.2.4.38 ETcVnHoughMethod

Offers Hough methods.

Syntax

Definition:

```
TYPE ETcVnHoughMethod :
(
    TCVN_HM_STANDARD      := 0,
    TCVN_HM_PROBABILISTIC := 1,
    TCVN_HM_MULTI_SCALE   := 2,
    TCVN_HM_GRADIENT      := 3,
    TCVN_HM_GRADIENT_ALT  := 4
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_HM_STANDARD	Classical or standard Hough transform. Every line is represented by two floating-point numbers (f1, f2), where f1 is a distance between (0,0) origin and the line, and f2 is the angle between x-axis and the normal to the line.
TCVN_HM_PROBABILISTIC	Probabilistic Hough transform. More efficient for pictures containing long linear segments. Returns line segments rather than the whole line, while each segment is represented by start and end point.
TCVN_HM_MULTI_SCALE	Multi scale variant of classical Hough transform.
TCVN_HM_GRADIENT	Method designed to find circles. See paper: HK Yuen, John Princen, John Illingworth, and Josef Kittler. Comparative study of hough transform methods for circle finding. Image and Vision Computing, 8(1):71–77, 1990
TCVN_HM_GRADIENT_ALT	Variation of the GRADIENT method.

Verwandte Funktionen

- F_VN_HoughCirclesExp [▶ 1222]

6.1.2.4.39 ETcVnInitializableFunction

Offers initializable functions.

Syntax

Definition:

```
TYPE ETcVnInitializableFunction :
(
    TCVN_IF_OCR := 0,
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_IF_OCR	F_VN_OCR

6.1.2.4.40 ETcVnInterpolationType

Offers interpolation types.

Syntax

Definition:

```
TYPE ETcVnInterpolationType :
(
    TCVN_IT_NEAREST_NEIGHBOR := 0,
    TCVN_IT_BILINEAR         := 1,
    TCVN_IT_BICUBIC          := 2,
    TCVN_IT_AREA_BASED       := 3,
    TCVN_IT_LANCZOS4         := 4
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_IT_NEAREST_NEIGHBOR	Nearest neighbor interpolation
TCVN_IT_BILINEAR	Bilinear interpolation
TCVN_IT_BICUBIC	Bicubic interpolation
TCVN_IT_AREA_BASED	Area based interpolation
TCVN_IT_LANCZOS4	Lanczos4 interpolation

Verwandte Funktionen

- F_VN_MatchTemplateAndEvaluateExp [▶ 1230]
- F_VN_RemapImageToLogPolarSpaceExp [▶ 1183]
- F_VN_RemapImageToPolarSpaceExp [▶ 1186]
- F_VN_ResizeImage [▶ 1188]
- F_VN_WarpAffineExp [▶ 1198]
- F_VN_WarpPerspectiveExp [▶ 1203]

6.1.2.4.41 ETcVnKeypointDetectionTypeAGAST

Offers different neighborhood types for AGAST method (For details see paper: E. Mair et al.: Adaptive and Generic Corner Detection Based on the Accelerated Segment Test, 2010).

Syntax

Definition:

```
TYPE ETcVnKeypointDetectionTypeAGAST :
(
    TCVN_KDT_AGAST_5_8   := 0,
    TCVN_KDT_AGAST_7_12d := 1,
    TCVN_KDT_AGAST_7_12s := 2,
    TCVN_KDT_AGAST_9_16  := 3
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_KDT_AGAST_5_8	5 of 8
TCVN_KDT_AGAST_7_12d	7 of 12 diamond
TCVN_KDT_AGAST_7_12s	7 of 12 square
TCVN_KDT_AGAST_9_16	9 of 16

Weiterführende Informationen

Das Enum `ETcVnKeypointDetectionTypeAGAST` wird in der Struktur TcVnParamsAGAST [▶ 246] verwendet.

6.1.2.4.42 ETcVnKeypointDetectionTypeFAST

Offers different neighborhood types for FAST method (For details see paper: E. Rosten: Machine Learning for High-speed Corner Detection, 2006).

Syntax

Definition:

```
TYPE ETcVnKeypointDetectionTypeFAST :
(
    TCVN_KDT_FAST_5_8  := 0,
    TCVN_KDT_FAST_7_12 := 1,
    TCVN_KDT_FAST_9_16 := 2
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_KDT_FAST_5_8	5 of 8
TCVN_KDT_FAST_7_12	7 of 12
TCVN_KDT_FAST_9_16	9 of 16

Weiterführende Informationen

Das Enum `ETcVnKeypointDetectionTypeFAST` wird in der Struktur TcVnParamsFAST [▶ 252] verwendet.

6.1.2.4.43 ETcVnKnn

Offers kNN model types.

Syntax

Definition:

```
TYPE ETcVnKnn :
(
    TCVN_KNN_CLASSIFIER       := 1,
    TCVN_KNN_NOVELTY_DETECTOR := 2,
    TCVN_KNN_REGRESSOR        := 4
)UDINT;
END_TYPE
```

Values

Name	Description
TCVN_KNN_CLASSIFIER	Classifier
TCVN_KNN_NOVELTY_DETECTOR	Novelty detector
TCVN_KNN_REGRESSOR	Regressor

6.1.2.4.44 ETcVnLineType

Offers line types.

Syntax

Definition:

```
TYPE ETcVnLineType :
(
    TCVN_LT_4_CONNECTED := 4,
    TCVN_LT_8_CONNECTED := 8,
    TCVN_LT_ANTIALIASED := 16
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_LT_4_CONNECTED	4-connected (pixels are connected horizontally and vertically).
TCVN_LT_8_CONNECTED	8-connected (pixels are connected horizontally, vertically and diagonally).
TCVN_LT_ANTIALIASED	Antialiased (drawn using Gaussian filtering, only implemented for 8-bit images).

Verwandte Funktionen

- F_VN_DrawCircleExp [▶ 1032]
- F_VN_DrawCirclesExp [▶ 1034]
- F_VN_DrawCircularArcExp [▶ 1036]
- F_VN_DrawContoursExp [▶ 1041]
- F_VN_DrawEllipseExp [▶ 1044]
- F_VN_DrawLineExp [▶ 1051]
- F_VN_DrawLineExp_TcVnVector4_DINT [▶ 1052]
- F_VN_DrawLineExp_TcVnVector4_LREAL [▶ 1053]
- F_VN_DrawLinesExp [▶ 1056]
- F_VN_DrawPointExp [▶ 1063]
- F_VN_DrawPointsExp [▶ 1065]
- F_VN_DrawRotatedRectangleExp [▶ 1072]
- F_VN_PutLabelExp [▶ 1083]
- F_VN_PutTextExp [▶ 1086]

6.1.2.4.45 ETcVnMorphologicalOperator

Offers morphological operators.

Syntax

Definition:

```
TYPE ETcVnMorphologicalOperator :
(
    TCVN_MO_EROSION                         := 0x0,
    TCVN_MO_DILATION                        := 0x1,
    TCVN_MO_OPENING                         := 0x2,
    TCVN_MO_CLOSING                         := 0x3,
    TCVN_MO_GRADIENT                        := 0x4,
    TCVN_MO_WHITE_TOPHAT                    := 0x5,
    TCVN_MO_BLACK_TOPHAT                    := 0x6,
    TCVN_MO_OPENING_BY_RECONSTRUCTION       := 0x40000002,
    TCVN_MO_CLOSING_BY_RECONSTRUCTION       := 0x40000003,
    TCVN_MO_WHITE_TOPHAT_BY_RECONSTRUCTION  := 0x40000005,
    TCVN_MO_BLACK_TOPHAT_BY_RECONSTRUCTION  := 0x40000006
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_MO_EROSION	Shrinks objects (regions of foreground (i.e. white) pixels), removing regions smaller than the structuring element completely.
TCVN_MO_DILATION	Expands objects and thereby closes small holes inside objects.
TCVN_MO_OPENING	Applies an erosion first, a dilation second. Objects smaller than the structuring element are removed while outer shapes remain largely the same.
TCVN_MO_CLOSING	Applies a dilation first, an erosion second. Holes inside objects that fit into the structuring element are closed completely while outer shapes remain largely the same.
TCVN_MO_GRADIENT	Difference between the dilation and the erosion of an image.
TCVN_MO_WHITE_TOPHAT	Difference between an input image and its opening.
TCVN_MO_BLACK_TOPHAT	Difference between an input image and its closing.
TCVN_MO_OPENING_BY_RECONSTRUCTION	Opening with subsequent reconstruction of objects that were not removed by the opening.
TCVN_MO_CLOSING_BY_RECONSTRUCTION	Closing with subsequent reconstruction of objects that were not removed by the closing.
TCVN_MO_WHITE_TOPHAT_BY_RECONSTRUCTION	White tophat with subsequent reconstruction of objects that were not removed by the white tophat.
TCVN_MO_BLACK_TOPHAT_BY_RECONSTRUCTION	Black tophat with subsequent reconstruction of objects that were not removed by the black tophat.

Verwandte Funktionen

- F_VN_MorphologicalOperator [▶ 1338]

6.1.2.4.46 ETcVnNbc

Offers normal Bayes classifier types.

Syntax

Definition:

```
TYPE ETcVnNbc :
(
    TCVN_NBC_CLASSIFIER       := 1,
    TCVN_NBC_NOVELTY_DETECTOR := 2
)UDINT;
END_TYPE
```

Values

Name	Description
TCVN_NBC_CLASSIFIER	Classifier
TCVN_NBC_NOVELTY_DETECTOR	Novelty detector

6.1.2.4.47 ETcVnNormalizationType

Offers normalization types.

Syntax

Definition:

```
TYPE ETcVnNormalizationType :
(
    TCVN_NT_INF               := 1,
    TCVN_NT_L1                := 2,
    TCVN_NT_L2                := 4,
    TCVN_NT_L2SQR             := 5,
    TCVN_NT_HAMMING           := 6,
    TCVN_NT_HAMMING2          := 7,
    TCVN_NT_RELATIVE_INF      := 9,
    TCVN_NT_RELATIVE_L1       := 10,
    TCVN_NT_RELATIVE_L2       := 12,
    TCVN_NT_RELATIVE_L2SQR    := 13,
    TCVN_NT_RELATIVE_HAMMING  := 14,
    TCVN_NT_RELATIVE_HAMMING2 := 15,
    TCVN_NT_MINMAX            := 32
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_NT_INF	Infinity norm
TCVN_NT_L1	L1 norm
TCVN_NT_L2	L2 (euclidean) norm
TCVN_NT_L2SQR	Squared L2 norm
TCVN_NT_HAMMING	Hamming distance (bitwise)
TCVN_NT_HAMMING2	Hamming distance (each 2 bits combined to one)
TCVN_NT_RELATIVE_INF	Infinity norm (when comparing 2 arrays, the norm of their difference is divided by the norm of the second array)
TCVN_NT_RELATIVE_L1	L1 norm (when comparing 2 arrays, the norm of their difference is divided by the norm of the second array)
TCVN_NT_RELATIVE_L2	L2 (euclidean) norm (when comparing 2 arrays, the norm of their difference is divided by the norm of the second array)
TCVN_NT_RELATIVE_L2SQR	Squared L2 norm (when comparing 2 arrays, the norm of their difference is divided by the norm of the second array)
TCVN_NT_RELATIVE_HAMMING	Hamming distance (bitwise; when comparing 2 arrays, the norm of their difference is divided by the norm of the second array)
TCVN_NT_RELATIVE_HAMMING2	Hamming distance (each 2 bits combined to one; when comparing 2 arrays, the norm of their difference is divided by the norm of the second array)
TCVN_NT_MINMAX	Normalize the values to a range given by a minimum and a maximum value.

Verwandte Funktionen

- F_VN_FindReferenceKeyPointsInImageORBExp [▶ 1383]
- F_VN_MatchDescriptorsBFExp [▶ 1404]
- F_VN_MatchDescriptorsKnnBFExp [▶ 1408]
- F_VN_NormalizeImageExp [▶ 1281]

6.1.2.4.48 ETcVnOcrModelType

Offers different OCR model types

Syntax

Definition:

```
TYPE ETcVnOcrModelType :
(
    TCVN_OMT_NUMBERS              := 580542139465730,
    TCVN_OMT_NUMBERS_SC           := 580542139465732,
    TCVN_OMT_UCLETTERS            := 580542139465736,
    TCVN_OMT_NUMBERS_SC_UCLETTERS := 580542139465744
)ULINT;
END_TYPE
```

Values

Name	Description
TCVN_OMT_NUMBERS	Classify numbers
TCVN_OMT_NUMBERS_SC	Classify numbers and special characters
TCVN_OMT_UCLETTERS	Classify uppercase letters
TCVN_OMT_NUMBERS_SC_UCLETTERS	Classify numbers, special characters and uppercase letters

6.1.2.4.49 ETcVnOcrOptions

Provides different options to indicate whether certain actions must be performed to achieve the intended result.

Syntax

Definition:

```
TYPE ETcVnOcrOptions :
(
    TCVN_OO_NONE       := 0,
    TCVN_OO_WITHBLANKS := 1
)UDINT;
END_TYPE
```

Values

Name	Description
TCVN_OO_NONE	No options
TCVN_OO_WITHBLANKS	Indicates that if blanks were found, they should be included in the result. By default, all blanks are omitted from the result.

6.1.2.4.50 ETcVnOrientationMethod

Offers methods to calculate the orientation of a set of points.

Syntax

Definition:

```
TYPE ETcVnOrientationMethod :
(
    TCVN_OM_PCA           := 0,
    TCVN_OM_FITELLIPSE    := 1,
    TCVN_OM_MOMENTS       := 2,
    TCVN_OM_ENCLOSINGRECT := 3
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_OM_PCA	Apply the PCA on the points and calculate the mean point as the center point and the rotation angle of the main principal axis
TCVN_OM_FITELLIPSE	Fit an ellipse around the points and calculate the center point and the rotation angle of the main axis
TCVN_OM_MOMENTS	Calculate the moments of the points and estimate the center and the rotation angle
TCVN_OM_ENCLOSINGRECT	Calculate a minimum area rectangle enclosing the points and calculate the center point and the rotation angle of the rectangle

6.1.2.4.51 ETcVnPaddingMode

Offers padding modes.

Syntax

Definition:

```
TYPE ETcVnPaddingMode :
(
    TCVN_PM_NONE        := 0,
    TCVN_PM_CROP_CENTER := 1,
    TCVN_PM_LETTERBOX   := 2
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_PM_NONE	Resize the image to the desired size without extra processing
TCVN_PM_CROP_CENTER	Resize and crop the image
TCVN_PM_LETTERBOX	Resize the image to the desired size while preserving the aspect ratio of the original image

6.1.2.4.52 ETcVnPixelConnectivity

Offers pixel connectivities.

Syntax

Definition:

```
TYPE ETcVnPixelConnectivity :
(
    TCVN_PC_4 := 4,
    TCVN_PC_8 := 8
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_PC_4	4-way
TCVN_PC_8	8-way

6.1.2.4.53 ETcVnPixelEncoding

Offers pixel encodings.

Syntax

Definition:

```
TYPE ETcVnPixelEncoding :
(
    TCVN_PE_NONE               := 0,
    TCVN_PE_BAYER_GR           := 1,
    TCVN_PE_BAYER_RG           := 2,
    TCVN_PE_BAYER_GB           := 3,
    TCVN_PE_BAYER_BG           := 4,
    TCVN_PE_YUV_411_UYYVYY     := 5,
    TCVN_PE_YUV_422_UYVY       := 6,
    TCVN_PE_YUV_422_YUYV       := 7,
    TCVN_PE_YCBCR_411_CBYYCRYY := 8,
    TCVN_PE_YCBCR_422_CBYCRY   := 9,
    TCVN_PE_YCBCR_422_YCBYCR   := 10
)BYTE;
END_TYPE
```

Values

Name	Description
TCVN_PE_NONE	No encoding available, i.e. every pixel value is independent of other pixels.
TCVN_PE_BAYER_GR	Pixels are encoded as a BayerGR pattern.
TCVN_PE_BAYER_RG	Pixels are encoded as a BayerRG pattern.
TCVN_PE_BAYER_GB	Pixels are encoded as a BayerGB pattern.
TCVN_PE_BAYER_BG	Pixels are encoded as a BayerBG pattern.
TCVN_PE_YUV_411_UYYVYY	Pixels are encoded as YUV411 (UYYVYY).
TCVN_PE_YUV_422_UYVY	Pixels are encoded as YUV422 (UYVY).
TCVN_PE_YUV_422_YUYV	Pixels are encoded as YUV422 (YUYV).
TCVN_PE_YCBCR_411_CBYYCRYY	Pixels are encoded as YCbCr411 (CbYYCrYY).
TCVN_PE_YCBCR_422_CBYCRY	Pixels are encoded as YCbCr422 (CbYCrY).
TCVN_PE_YCBCR_422_YCBYCR	Pixels are encoded as YCbCr422 (YCbYCr).

Weiterführende Informationen

Das Enum `ETcVnPixelEncoding` wird in der Struktur TcVnPixelFormat [▶ 256] verwendet. Durch Abfragen der Pixelformatinformationen eines Bildes erhält man so Auskunft über das Pixel-Encoding. Häufiger Anwendungsfall ist die Abfrage auf ein Bayer Pattern, um herauszufinden ob ein Bild z. B. noch in ein RGB Format konvertiert werden muss.

6.1.2.4.54 ETcVnPixelPackMode

Offers pixel packing modes.

Syntax

Definition:

```
TYPE ETcVnPixelPackMode :
(
    TCVN_PPM_NONE          := 0,
    TCVN_PPM_MONO1P        := 1,
    TCVN_PPM_MONO2P        := 2,
    TCVN_PPM_MONO4P        := 3,
    TCVN_PPM_MONO10PACKED  := 4,
    TCVN_PPM_MONO12PACKED  := 5,
    TCVN_PPM_RGB10V1PACKED := 6,
    TCVN_PPM_RGB10P32      := 7,
    TCVN_PPM_RGB12V1PACKED := 8,
```

```
    TCVN_PPM_RGB565P        := 9,
    TCVN_PPM_BGR565P        := 10,
    TCVN_PPM_MONO10P        := 11,
    TCVN_PPM_MONO12P        := 12,
    TCVN_PPM_MONO14P        := 13
)BYTE;
END_TYPE
```

Values

Name	Description
TCVN_PPM_NONE	No packing
TCVN_PPM_MONO1P	Mono1p
TCVN_PPM_MONO2P	Mono2p
TCVN_PPM_MONO4P	Mono4p
TCVN_PPM_MONO10PACKED	Mono10Packed or BayerXX10Packed
TCVN_PPM_MONO12PACKED	Mono12Packed or BayerXX12Packed
TCVN_PPM_RGB10V1PACKED	RGB10V1Packed
TCVN_PPM_RGB10P32	RGB10V2Packed (RGB10p32)
TCVN_PPM_RGB12V1PACKED	RGB12V1Packed
TCVN_PPM_RGB565P	RGB565p
TCVN_PPM_BGR565P	BGR565p
TCVN_PPM_MONO10P	Mono10p or BayerXX10p
TCVN_PPM_MONO12P	Mono12p or BayerXX12p
TCVN_PPM_MONO14P	Mono14p or BayerXX14p

Weiterführende Informationen

Das Enum `ETcVnPixelPackMode` wird in der Struktur TcVnPixelFormat [▶ 256] verwendet.

6.1.2.4.55 ETcVnPrototypeClusterer

Offers prototype clusterer model types.

Syntax

Definition:

```
TYPE ETcVnPrototypeClusterer :
(
    TCVN_PC_NOVELTY_DETECTOR := 2,
    TCVN_PC_CLUSTERER        := 8
)UDINT;
END_TYPE
```

Values

Name	Description
TCVN_PC_NOVELTY_DETECTOR	Novelty detector
TCVN_PC_CLUSTERER	Clusterer

6.1.2.4.56 ETcVnRectangleIntersection

Offers rectangle intersection types.

Syntax

Definition:

```
TYPE ETcVnRectangleIntersection :
(
    TCVN_RI_NONE    := 0,
    TCVN_RI_PARTIAL := 1,
    TCVN_RI_FULL    := 2
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_RI_NONE	No intersection
TCVN_RI_PARTIAL	Partial intersection
TCVN_RI_FULL	Full intersection

6.1.2.4.57 ETcVnRotationAngle

Offers rotation angles.

Syntax

Definition:

```
TYPE ETcVnRotationAngle :
(
    TCVN_RA_90_DEG  := 0,
    TCVN_RA_180_DEG := 1,
    TCVN_RA_270_DEG := 2
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_RA_90_DEG	90 degrees
TCVN_RA_180_DEG	180 degrees
TCVN_RA_270_DEG	270 degrees

Verwandte Funktionen

- F_VN_RotateImage [▶ 1191]

6.1.2.4.58 ETcVnRTrees

Offers RTrees model types.

Syntax

Definition:

```
TYPE ETcVnRTrees :
(
    TCVN_RT_CLASSIFIER := 1,
    TCVN_RT_REGRESSOR  := 4
)UDINT;
END_TYPE
```

Values

Name	Description
TCVN_RT_CLASSIFIER	Classifier
TCVN_RT_REGRESSOR	Regressor

6.1.2.4.59 ETcVnSignedNormalization

Offers options for normalizing images with signed data types.

Syntax

Definition:

```
TYPE ETcVnSignedNormalization :
(
    TCVN_SN_FIX_ZERO   := 0,
    TCVN_SN_FULL_SCALE := 1
)INT;
END_TYPE
```

Values

Name	Description
TCVN_SN_FIX_ZERO	The value 0 is fixed, i.e. the normalized image might only reach either the minimum or maximum value.
TCVN_SN_FULL_SCALE	The image is normalized to the full value range, i.e. a prior value of 0 might be != 0 after normalization.

Verwandte Funktionen

- F_VN_NormalizeImageForDisplay [▶ 1282]

6.1.2.4.60 ETcVnSolvePnPMethod

Offers SolvePnP methods.

Syntax

Definition:

```
TYPE ETcVnSolvePnPMethod :
(
    TCVN_SPM_ITERATIVE   := 0,
    TCVN_SPM_EPNP        := 1,
    TCVN_SPM_P3P         := 2,
    TCVN_SPM_AP3P        := 5,
    TCVN_SPM_IPPE        := 6,
    TCVN_SPM_IPPE_SQUARE := 7,
    TCVN_SPM_SQPNP       := 8
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_SPM_ITERATIVE	Levenberg-Marquardt optimization. Requires at least 4 planar or 6 non-planar reference points.
TCVN_SPM_EPNP	Efficient perspective-n-point camera pose estimation. Requires at least 4 reference points.
TCVN_SPM_P3P	Complete solution classification for the perspective-three-point problem. Requires exactly 4 reference points to find a unique solution.
TCVN_SPM_AP3P	An efficient algebraic solution to the perspective-three-point problem. Requires exactly 4 reference points to find a unique solution.
TCVN_SPM_IPPE	Infinitesimal plane-based pose estimation. Requires at least 4 coplanar reference points.
TCVN_SPM_IPPE_SQUARE	IPPE specialization for squares. Requires exactly 4 reference points in the following order (hsl: half square length): (-hsl, hsl, 0), (hsl, hsl, 0), (hsl, -hsl, 0), (-hsl, -hsl, 0)
TCVN_SPM_SQPNP	A consistently fast and globally optimal solution to the perspective-n-point problem. Requires at least 4 reference points.

6.1.2.4.61 ETcVnSta

Offers Simplified TopoART model types.

Syntax

Definition:

```
TYPE ETcVnSta :
(
    TCVN_STA_CLASSIFIER       := 1,
    TCVN_STA_NOVELTY_DETECTOR := 2,
    TCVN_STA_REGRESSOR        := 4,
    TCVN_STA_CLUSTERER        := 8
)UDINT;
END_TYPE
```

Values

Name	Description
TCVN_STA_CLASSIFIER	Classifier
TCVN_STA_NOVELTY_DETECTOR	Novelty detector
TCVN_STA_REGRESSOR	Regressor
TCVN_STA_CLUSTERER	Clusterer

6.1.2.4.62 ETcVnStructuringElementShape

Offers shapes for a structuring element.

Syntax

Definition:

```
TYPE ETcVnStructuringElementShape :
(
    TCVN_SES_RECTANGLE := 0,
    TCVN_SES_CROSS     := 1,
    TCVN_SES_ELLIPSE   := 2
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_SES_RECTANGLE	Rectangle
TCVN_SES_CROSS	Cross
TCVN_SES_ELLIPSE	Ellipse

Verwandte Funktionen

- F_VN_CreateStructuringElement [▶ 1314]

6.1.2.4.63 ETcVnSvm

Offers SVM model types.

Syntax

Definition:

```
TYPE ETcVnSvm :
(
    TCVN_SVM_C_CLASSIFIER     := 1,
    TCVN_SVM_NU_CLASSIFIER    := 65537,
    TCVN_SVM_NOVELTY_DETECTOR := 2,
    TCVN_SVM_EPS_REGRESSOR    := 4,
    TCVN_SVM_NU_REGRESSOR     := 65540
)UDINT;
END_TYPE
```

Values

Name	Description
TCVN_SVM_C_CLASSIFIER	Classifier
TCVN_SVM_NU_CLASSIFIER	Classifier
TCVN_SVM_NOVELTY_DETECTOR	Novelty detector
TCVN_SVM_EPS_REGRESSOR	Regressor
TCVN_SVM_NU_REGRESSOR	Regressor

6.1.2.4.64 ETcVnSvmKernelType

Offers SVM kernel types.

Syntax

Definition:

```
TYPE ETcVnSvmKernelType :
(
    TCVN_SKT_LINEAR  := 0,
    TCVN_SKT_POLY    := 1,
    TCVN_SKT_RBF     := 2,
    TCVN_SKT_SIGMOID := 3,
    TCVN_SKT_CHI2    := 4,
    TCVN_SKT_INTER   := 5
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_SKT_LINEAR	Linear kernel (fast)
TCVN_SKT_POLY	Polynomial kernel
TCVN_SKT_RBF	Radial basis function (RBF) kernel (good default)
TCVN_SKT_SIGMOID	Sigmoid kernel
TCVN_SKT_CHI2	Chi-squared kernel
TCVN_SKT_INTER	Histogram intersection kernel

6.1.2.4.65 ETcVnSvmSgdClassifierMarginType

Offers different margin types for SVMSGD classifiers.

Syntax

Definition:

```
TYPE ETcVnSvmSgdClassifierMarginType :
(
    TCVN_SSCMT_SOFT_MARGIN := 0,
    TCVN_SSCMT_HARD_MARGIN := 1
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_SSCMT_SOFT_MARGIN	Soft margin allowing outliers. (beneficial for classes that cannot be separated linearly)
TCVN_SSCMT_HARD_MARGIN	Hard margin. (best-suited for linearly separable classes)

6.1.2.4.66 ETcVnSvmSgdClassifierType

Offers different types of SVMSGD classifiers.

Syntax

Definition:

```
TYPE ETcVnSvmSgdClassifierType :
(
    TCVN_SSCT_SGD  := 0,
    TCVN_SSCT_ASGD := 1
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_SSCT_SGD	Stochastic Gradient Descent
TCVN_SSCT_ASGD	Average Stochastic Gradient Descent

6.1.2.4.67 ETcVnTemplateMatchMethod

Offers methods for template matching.

Syntax

Definition:

```
TYPE ETcVnTemplateMatchMethod :
(
    TCVN_TMM_SQDIFF        := 0,
    TCVN_TMM_SQDIFF_NORMED := 1,
    TCVN_TMM_CCORR         := 2,
    TCVN_TMM_CCORR_NORMED  := 3,
    TCVN_TMM_CCOEFF        := 4,
    TCVN_TMM_CCOEFF_NORMED := 5
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_TMM_SQDIFF	Squared difference (supports template mask)
TCVN_TMM_SQDIFF_NORMED	Normalized squared difference
TCVN_TMM_CCORR	Cross-correlation
TCVN_TMM_CCORR_NORMED	Normalized cross-correlation (supports template mask)
TCVN_TMM_CCOEFF	Correlation coefficient
TCVN_TMM_CCOEFF_NORMED	Normalized correlation coefficient

Verwandte Funktionen

- F_VN_MatchTemplateAndEvaluateExp [▶ 1230]
- F_VN_MatchTemplateExp [▶ 1232]

6.1.2.4.68 ETcVnThresholdType

Offers threshold types.

Syntax

Definition:

```
TYPE ETcVnThresholdType :
(
    TCVN_TT_BINARY              := 0,
    TCVN_TT_BINARY_INV          := 1,
    TCVN_TT_TRUNC               := 2,
    TCVN_TT_TOZERO              := 3,
    TCVN_TT_TOZERO_INV          := 4,
    TCVN_TT_OTSU_BINARY         := 8,
    TCVN_TT_OTSU_BINARY_INV     := 9,
    TCVN_TT_OTSU_TRUNC          := 10,
    TCVN_TT_OTSU_TOZERO         := 11,
    TCVN_TT_OTSU_TOZERO_INV     := 12,
    TCVN_TT_TRIANGLE_BINARY     := 16,
    TCVN_TT_TRIANGLE_BINARY_INV := 17,
    TCVN_TT_TRIANGLE_TRUNC      := 18,
    TCVN_TT_TRIANGLE_TOZERO     := 19,
    TCVN_TT_TRIANGLE_TOZERO_INV := 20
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_TT_BINARY	Binary threshold
TCVN_TT_BINARY_INV	Inverted binary threshold
TCVN_TT_TRUNC	Truncated threshold (pixels > thresh are set to thresh, others keep their value)
TCVN_TT_TOZERO	To zero threshold (pixels < thresh are set to zero, others keep their value)
TCVN_TT_TOZERO_INV	Inverted to zero threshold (pixels > thresh are set to zero, others keep their value)
TCVN_TT_OTSU_BINARY	Binary threshold with the threshold value selected according to the Otsu algorithm
TCVN_TT_OTSU_BINARY_INV	Inverted binary threshold with the threshold value selected according to the Otsu algorithm
TCVN_TT_OTSU_TRUNC	Truncated threshold with the threshold value selected according to the Otsu algorithm
TCVN_TT_OTSU_TOZERO	To zero threshold with the threshold value selected according to the Otsu algorithm
TCVN_TT_OTSU_TOZERO_INV	Inverted to zero threshold with the threshold value selected according to the Otsu algorithm
TCVN_TT_TRIANGLE_BINARY	Binary threshold with the threshold value selected according to the Triangle algorithm
TCVN_TT_TRIANGLE_BINARY_INV	Inverted binary threshold with the threshold value selected according to the Triangle algorithm
TCVN_TT_TRIANGLE_TRUNC	Truncated threshold with the threshold value selected according to the Triangle algorithm
TCVN_TT_TRIANGLE_TOZERO	To zero threshold with the threshold value selected according to the Triangle algorithm
TCVN_TT_TRIANGLE_TOZERO_INV	Inverted to zero threshold with the threshold value selected according to the Triangle algorithm

Weiterführende Informationen

Das Enum `ETcVnThresholdType` bietet die Auswahl der implementierten Methoden zur Schwellwertberechnung an. Es wird ebenfalls in der Struktur TcVnParamsBlobDetection [▶ 247] verwendet.

Verwandte Funktionen

- F_VN_AdaptiveThresholdExp [▶ 1358]
- F_VN_Threshold [▶ 1363]

6.1.2.4.69 ETcVnTimestamp

Offers image acquisition timestamps.

Syntax

Definition:

```
TYPE ETcVnTimestamp :
(
    TCVN_TS_IMAGE_COMPLETED := 1
)ULINT;
END_TYPE
```

Values

Name	Description
TCVN_TS_IMAGE_COMPLETED	Timestamp when the image was completed, i.e. all packets were received by the GigEVision driver.

6.1.2.4.70 ETcVnVectorCompareMethod

Offers methods for vector comparison.

Syntax

Definition:

```
TYPE ETcVnVectorCompareMethod :
(
    TCVN_VCM_EUCLIDEAN    := 0,
    TCVN_VCM_ELEMENTWISE := 1
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_VCM_EUCLIDEAN	Euclidean
TCVN_VCM_ELEMENTWISE	Elementwise

Verwandte Funktionen

- F_VN_MaxPixelValueExp [▶ 1257]
- F_VN_MinPixelValueExp [▶ 1259]

6.1.2.4.71 ETcVnWorldCoordinateSystem

Offers world coordinate system orientations.

Syntax

Definition:

```
TYPE ETcVnWorldCoordinateSystem :
(
    TCVN_WCS_X_RIGHT_Y_DOWN := 0,
    TCVN_WCS_X_DOWN_Y_RIGHT := 1,
    TCVN_WCS_X_LEFT_Y_DOWN  := 2,
    TCVN_WCS_X_UP_Y_RIGHT   := 3,
    TCVN_WCS_X_RIGHT_Y_UP   := 4,
    TCVN_WCS_X_DOWN_Y_LEFT  := 5,
    TCVN_WCS_X_LEFT_Y_UP    := 6,
    TCVN_WCS_X_UP_Y_LEFT    := 7
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_WCS_X_RIGHT_Y_DOWN	X axis points right, Y axis down (same orientation as image coordinate system)
TCVN_WCS_X_DOWN_Y_RIGHT	X axis points down, Y axis right
TCVN_WCS_X_LEFT_Y_DOWN	X axis points left, Y axis down
TCVN_WCS_X_UP_Y_RIGHT	X axis points up, Y axis right
TCVN_WCS_X_RIGHT_Y_UP	X axis points right, Y axis up
TCVN_WCS_X_DOWN_Y_LEFT	X axis points down, Y axis left
TCVN_WCS_X_LEFT_Y_UP	X axis points left, Y axis up
TCVN_WCS_X_UP_Y_LEFT	X axis points up, Y axis left

6.1.2.4.72 ETcWatchdogAccumulationType

Offers watchdog accumulation types to compute the fraction processed.

Syntax

Definition:

```
TYPE ETcWatchdogAccumulationType :
(
    WD_ACC_TYPE_MEAN    := 1,
    WD_ACC_TYPE_PRODUCT := 2
)UINT;
END_TYPE
```

Values

Name	Description
WD_ACC_TYPE_MEAN	Calculates the mean of the individual fractions processed over all monitored functions (recommended for independent functions).
WD_ACC_TYPE_PRODUCT	Calculates the product of the individual fractions processed over all monitored functions (recommended for dependent functions).

Weiterführende Informationen

Mittels dieses Enums lässt sich einstellen, wie ein Watchdog [▶ 156] den Anteil der ausgeführten Funktionen (`nFractionProcessed`) bei einer Aneinanderreihung mehrerer Funktionen berechnet.

Als Beispiel zur Verdeutlichung der Funktionsweise wird angenommen, dass drei Funktionen überwacht wurden. Die erste wurde zu 100% ausgeführt, die zweite zu 50% und die dritte zu 0%.

WD_ACC_TYPE_MEAN	Der Durchschnittswert der Ausführungsanteile aller Funktionen wird als Ergebnis zurückgegeben. In dem Beispielfall würde das Ergebnis sein: **(100% + 50% + 0%) / 3 = 50%**.
WD_ACC_TYPE_PRODUCT	Das Produkt der Ausführungsanteile aller Funktionen wird als Ergebnis zurückgegeben. In dem Beispielfall würde das Ergebnis sein: **100% * 50% * 0% = 0%**.

Die Anwendungsfälle unterscheiden sich je nach Programmstruktur und sicheren Zuständen der Maschine.

Verwandte Funktionen

- F_VN_StartAbsWatchdogExp [▶ 1021]
- F_VN_StartRelWatchdogExp [▶ 1023]

6.1.2.4.73 GEV_CAMERA_STATE

Describes the GigEVision camera connection state.

Syntax

Definition:

```
TYPE GEV_CAMERA_STATE :
(
    GEV_CAMERA_IDLE               := 0,
    GEV_CONTROL_CHANNEL_OPEN_MASK := 1,
    GEV_STREAM_CHANNELS_OPEN_MASK := 2,
    GEV_CONTROL_CHANNEL_OPEN      := 1,
    GEV_STREAM_CHANNELS_OPEN      := 3
)BYTE;
END_TYPE
```

Values

Name	Description
GEV_CAMERA_IDLE	Camera is idle (no control or stream channel open).
GEV_CONTROL_CHANNEL_OPEN_MASK	The control channel is open.
GEV_STREAM_CHANNELS_OPEN_MASK	At least 1 stream channel is open.
GEV_CONTROL_CHANNEL_OPEN	The control channel is open, but no stream channel.
GEV_STREAM_CHANNELS_OPEN	At least 1 stream channel and the control channel are open.

6.1.2.4.74 ETcVnDataLayout

Offers image data layouts.

Syntax

Definition:

```
TYPE ETcVnDataLayout :
(
    TCVN_DL_DEFAULT := 0x0,
    TCVN_DL_2D_HW   := 0x12,
    TCVN_DL_2D_WH   := 0x21,
    TCVN_DL_3D_CHW  := 0x123,
    TCVN_DL_3D_HWC  := 0x231,
    TCVN_DL_4D_NCHW := 0x1234,
    TCVN_DL_4D_NHWC := 0x1342
)ULINT;
END_TYPE
```

Values

Name	Description
TCVN_DL_DEFAULT	Default
TCVN_DL_2D_HW	2D data HW, multi/single channel image
TCVN_DL_2D_WH	2D data WH, multi/single channel image
TCVN_DL_3D_CHW	3D data CHW, single channel image
TCVN_DL_3D_HWC	3D data HWC, single channel image
TCVN_DL_4D_NCHW	4D data NCHW, single channel image
TCVN_DL_4D_NHWC	4D data NHWC, single channel image

6.1.2.4.75 ETcVnReduceType

Offers reduction types.

Syntax

Definition:

```
TYPE ETcVnReduceType :
(
    TCVN_RT_REDUCE_SUM := 0,
    TCVN_RT_REDUCE_AVG := 1,
    TCVN_RT_REDUCE_MAX := 2,
    TCVN_RT_REDUCE_MIN := 3
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_RT_REDUCE_SUM	The sum (column/row-wise) of all rows/columns
TCVN_RT_REDUCE_AVG	The mean (column/row-wise) of all rows/columns
TCVN_RT_REDUCE_MAX	The maximum (column/row-wise) of all rows/columns
TCVN_RT_REDUCE_MIN	The minimum (column/row-wise) of all rows/columns

6.1.2.4.76 ETcVnCircularityCalculationMethod

Offers methods to calculate contour circularity

Syntax

Definition:

```
TYPE ETcVnCircularityCalculationMethod :
(
    TCVN_CCM_DEFAULT                := 0,
    TCVN_CCM_CONVEXHULL_PERIMETER   := 1,
    TCVN_CCM_MAX_RADIUS             := 2,
    TCVN_CCM_ENCLOSINGCIRCLE_RADIUS := 3
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_CCM_DEFAULT	4pi * area / perimeter^2
TCVN_CCM_CONVEXHULL_PERIMETER	4pi * area / (convex hull perimeter)^2
TCVN_CCM_MAX_RADIUS	area / (pi * r^2), r = max distance to the contour's center of mass
TCVN_CCM_ENCLOSINGCIRCLE_RADIUS	area / (pi * r^2), r = radius of the minimum enclosing circle

6.1.2.4.77 ETcVnDataMatrixOptions

Offers Data Matrix code detection options.

Syntax

Definition:

```
TYPE ETcVnDataMatrixOptions :
(
    TCVN_DMO_DEFAULT          := 0,
    TCVN_DMO_BINARY_INPUT     := 1,
    TCVN_DMO_GLOBAL_THRESHOLD := 2,
    TCVN_DMO_BINARY_GRID      := 4,
    TCVN_DMO_STATIC_GRID      := 8,
    TCVN_DMO_REGULAR_COLOR    := 16,
    TCVN_DMO_INVERTED_COLOR   := 32
)ULINT;
END_TYPE
```

Values

Name	Description
TCVN_DMO_DEFAULT	Expects a graylevel image as input. Applies an adaptive local threshold and creates a dynamic grid based on the clock pattern.
TCVN_DMO_BINARY_INPUT	Expects a binary image as input (all pixels must contain 0 or 255, disables internal preprocessing)
TCVN_DMO_GLOBAL_THRESHOLD	Applies a global instead of an adaptive local threshold (requires uniform illumination)
TCVN_DMO_BINARY_GRID	Create the dynamic grid based on the binary instead of the graylevel image
TCVN_DMO_STATIC_GRID	Use an equally spaced static grid instead of a dynamic one
TCVN_DMO_REGULAR_COLOR	Only search for regular color (dark code on light background)
TCVN_DMO_INVERTED_COLOR	Only search for inverted color (light code on dark background)

6.1.2.4.78 ETcVnLayerNameType

Offer different types of neural network model layer names

Syntax

Definition:

```
TYPE ETcVnLayerNameType :
(
    TCVN_LNT_ALL     := 0,
    TCVN_LNT_OUTPUTS := 1,
    TCVN_LNT_INPUTS  := 2
)DINT;
END_TYPE
```

Values

Name	Description
TCVN_LNT_ALL	Return the name of all layers
TCVN_LNT_OUTPUTS	Return the names of unconnected output layers
TCVN_LNT_INPUTS	Return the names of model inputs

6.1.2.5 Structs

6.1.2.5.1 GVSP_IMAGE_INFO

Shows GVSP (GigE Vision Streaming Protocol) meta information.

Syntax

Definition:

```
TYPE GVSP_IMAGE_INFO :
STRUCT
    CameraIpAddress : UDINT;
    LocalIpAddress  : UDINT;
    CameraUdpPort   : UINT;
    LocalUdpPort    : UINT;
    GvspChannelId   : UINT;
    GevStatus       : UINT;
    BlockId         : ULINT;
```

```
    LeaderInfo      : GVSP_LEADER_PAYLOAD_IMAGE;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
CameraIpAddress	UDINT	Camera IP address
LocalIpAddress	UDINT	Local IP address
CameraUdpPort	UINT	Camera UDP port
LocalUdpPort	UINT	Local UDP port
GvspChannelId	UINT	ID of GVSP channel
GevStatus	UINT	GigE Vision status code
BlockId	ULINT	Block Id (incremented for each acquired image, but reset to 0 on overflow)
LeaderInfo	GVSP_LEADER_PAYLOAD_IMAGE [▶ 239]	Contains information about timestamp, pixel format and size.

6.1.2.5.2 GVSP_LEADER_PAYLOAD_IMAGE

Shows information over the GVSP (GigE Vision Streaming Protocol) leader payload image.

Syntax

Definition:

```
TYPE GVSP_LEADER_PAYLOAD_IMAGE :
STRUCT
    Timestamp   : ULINT;
    PixelFormat : GVSP_PIXEL_FORMAT;
    SizeX       : UDINT;
    SizeY       : UDINT;
    OffsetX     : UDINT;
    OffsetY     : UDINT;
    PaddingX    : UINT;
    PaddingY    : UINT;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
Timestamp	ULINT	Image timestamp
PixelFormat	GVSP_PIXEL_FORMAT [▶ 239]	Image pixel format
SizeX	UDINT	Image size in x direction
SizeY	UDINT	Image size in y direction
OffsetX	UDINT	Image x-offset from (0,0) origin
OffsetY	UDINT	Image y-offset from (0,0) origin
PaddingX	UINT	Image padding in x direction
PaddingY	UINT	Image padding in y direction

6.1.2.5.3 GVSP_PIXEL_FORMAT

Shows the GVSP (GigE Vision Streaming Protocol) pixel format.

Syntax

Definition:

```
TYPE GVSP_PIXEL_FORMAT :
STRUCT
    Color              : BYTE;
    EffectivePixelSize : BYTE;
    Id                 : UINT;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
Color	BYTE	Indicates if the pixel format is mono or color.
EffectivePixelSize	BYTE	Effective pixel size in bit
Id	UINT	GVSP pixel format ID

6.1.2.5.4 TcVnCameraCalibrationOptions

Offers camera calibration options.

Syntax

Definition:

```
TYPE TcVnCameraCalibrationOptions :
STRUCT
    bUseIntrinsicGuess : BIT;
    bFixAspectRatio    : BIT;
    bFixPrincipalPoint : BIT;
    bZeroTangentDist   : BIT;
    bFixFocalLength    : BIT;
    bFixK1             : BIT;
    bFixK2             : BIT;
    bFixK3             : BIT;
    bFixK4             : BIT;
    bFixK5             : BIT;
    bFixK6             : BIT;
    bRationalModel     : BIT;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
bUseIntrinsicGuess	BIT	The camera matrix and distortion coefficients already contain a valid initial guess, which is optimized further.
bFixAspectRatio	BIT	The ratio fx/fy stays the same as in the input camera matrix.
bFixPrincipalPoint	BIT	The principal point is fixed to the image center (or provided cx,cy if bUseIntrinsicGuess).
bZeroTangentDist	BIT	The tangential distortion coefficients (p1, p2) are forced to 0.
bFixFocalLength	BIT	The parameters fx and fy stay the same as in the input camera matrix.
bFixK1	BIT	The radial distortion coefficient k1 is fixed to 0 (or provided input if bUseIntrinsicGuess)
bFixK2	BIT	The radial distortion coefficient k2 is fixed to 0 (or provided input if bUseIntrinsicGuess)
bFixK3	BIT	The radial distortion coefficient k3 is fixed to 0 (or provided input if bUseIntrinsicGuess)
bFixK4	BIT	The radial distortion coefficient k4 is fixed to 0 (or provided input if bUseIntrinsicGuess)
bFixK5	BIT	The radial distortion coefficient k5 is fixed to 0 (or provided input if bUseIntrinsicGuess)
bFixK6	BIT	The radial distortion coefficient k6 is fixed to 0 (or provided input if bUseIntrinsicGuess)
bRationalModel	BIT	The radial distortion coefficients k4, k5, k6 are enabled.

6.1.2.5.5 TcVnCircularArc

Describes a circular arc.

Syntax

Definition:

```
TYPE TcVnCircularArc :
STRUCT
    aCenter     : TcVnPoint2_REAL;
    fRadius     : REAL;
    fStartAngle : REAL;
    fEndAngle   : REAL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
aCenter	TcVnPoint2_REAL [▶ 171]	Center of the circular arc
fRadius	REAL	Radius of the circular arc
fStartAngle	REAL	Start angle of the circular arc
fEndAngle	REAL	End angle of the circular arc

6.1.2.5.6 TcVnCodeGrades1D

Describes code quality grades for 1D barcodes according to ISO / IEC 15416:2016. The grades range from 0 (very bad) to 4 (very good) and are averaged over several scan lines.

Syntax

Definition:

```
TYPE TcVnCodeGrades1D :
STRUCT
    fDecode          : REAL;
    fSymbolContrast  : REAL;
    fMinReflectance  : REAL;
    fMinEdgeContrast : REAL;
    fModulation      : REAL;
    fDefects         : REAL;
    fDecodability    : REAL;
    fOverall         : REAL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
fDecode	REAL	Decode grade
fSymbolContrast	REAL	Symbol contrast grade
fMinReflectance	REAL	Minimum reflectance grade
fMinEdgeContrast	REAL	Minimum edge contrast grade
fModulation	REAL	Modulation grade
fDefects	REAL	Defects grade
fDecodability	REAL	Decodability grade
fOverall	REAL	Overall code grade, i.e. the minimum achieved individual grade averaged over all scan lines. If different scan lines lead to different decoded data, the overall grade is 0.

6.1.2.5.7 TcVnCodeGradesDM

Describes code quality grades for Data Matrix codes according to ISO / IEC 15415:2011. The grades range from 0 (very bad) to 4 (very good).

Syntax

Definition:

```
TYPE TcVnCodeGradesDM :
STRUCT
    nDecode               : USINT;
    nSymbolContrast       : USINT;
    nModulation           : USINT;
    nReflectanceMargin    : USINT;
    nFixedPatternDamage   : USINT;
```

```
    nAxialNonuniformity    : USINT;
    nGridNonuniformity     : USINT;
    nUnusedErrorCorrection : USINT;
    nOverall               : USINT;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
nDecode	USINT	Decode grade
nSymbolContrast	USINT	Symbol contrast grade
nModulation	USINT	Modulation grade
nReflectanceMargin	USINT	Reflectance margin grade
nFixedPatternDamage	USINT	Fixed pattern damage grade
nAxialNonuniformity	USINT	Axial nonuniformity grade
nGridNonuniformity	USINT	Grid nonuniformity grade
nUnusedErrorCorrection	USINT	Unused error correction grade
nOverall	USINT	Overall code grade, i.e. the minimum achieved individual grade

6.1.2.5.8 TcVnCodeGradesQR

Describes code quality grades for QR codes according to ISO / IEC 15415:2011. The grades range from 0 (very bad) to 4 (very good).

Syntax

Definition:

```
TYPE TcVnCodeGradesQR :
STRUCT
    nDecode                : USINT;
    nSymbolContrast        : USINT;
    nModulation            : USINT;
    nReflectanceMargin     : USINT;
    nFixedPatternDamage    : USINT;
    nAxialNonuniformity    : USINT;
    nGridNonuniformity     : USINT;
    nUnusedErrorCorrection : USINT;
    nFormatInfo            : USINT;
    nVersionInfo           : USINT;
    nOverall               : USINT;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
nDecode	USINT	Decode grade
nSymbolContrast	USINT	Symbol contrast grade
nModulation	USINT	Modulation grade
nReflectanceMargin	USINT	Reflectance margin grade
nFixedPatternDamage	USINT	Fixed pattern damage grade
nAxialNonuniformity	USINT	Axial nonuniformity grade
nGridNonuniformity	USINT	Grid nonuniformity grade
nUnusedErrorCorrection	USINT	Unused error correction grade
nFormatInfo	USINT	Format information grade
nVersionInfo	USINT	Version information grade
nOverall	USINT	Overall code grade, i.e. the minimum achieved individual grade

6.1.2.5.9 TcVnDMatch

Describes a descriptor match

Syntax

Definition:

```
TYPE TcVnDMatch :
STRUCT
    nQueryIdx : DINT;
    nTrainIdx : DINT;
    nImageIdx : DINT;
    fDistance : REAL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
nQueryIdx	DINT	Query descriptor index
nTrainIdx	DINT	Train descriptor index
nImageIdx	DINT	Train image index
fDistance	REAL	Distance between the descriptors (smaller distance means better match)

6.1.2.5.10 TcVnImageInfo

Shows image information.

Syntax

Definition:

```
TYPE TcVnImageInfo :
STRUCT
    nImageSize    : ULINT;
    nWidth        : UDINT;
    nHeight       : UDINT;
    nXPadding     : UINT;
    nYPadding     : UINT;
    stPixelFormat : TcVnPixelFormat;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
nImageSize	ULINT	Image size (number of pixels)
nWidth	UDINT	Image width
nHeight	UDINT	Image height
nXPadding	UINT	Image x-padding
nYPadding	UINT	Image y-padding
stPixelFormat	TcVnPixelFormat [▶ 256]	Pixel format

6.1.2.5.11 TcVnKeyPoint

Describes a key point.

Syntax

Definition:

```
TYPE TcVnKeyPoint :
STRUCT
    aPoint    : TcVnPoint2_REAL;
    fDiameter : REAL;
    fAngle    : REAL;
    fResponse : REAL;
    nOctave   : DINT;
    nClassId  : DINT;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
aPoint	TcVnPoint2_REAL [▶ 171]	Position
fDiameter	REAL	Diameter
fAngle	REAL	Angle
fResponse	REAL	Response
nOctave	DINT	Octave
nClassId	DINT	Class ID

6.1.2.5.12 TcVnMatrix

Offers a user-defined matrix with variable rows, columns and element-type.

Syntax

Definition:

```
TYPE TcVnMatrix :
STRUCT
    nRows : UDINT;
    nCols : UDINT;
    eType : ETcVnElementType;
    pData : PVOID;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
nRows	UDINT	Number of rows
nCols	UDINT	Number of columns
eType	ETcVnElementType [▶ 210]	Element type
pData	PVOID	Pointer to the data

6.1.2.5.13 TcVnMoments

Offers image or contour moments.

Syntax

Definition:

```
TYPE TcVnMoments :
STRUCT
    fM00  : LREAL;
    fM10  : LREAL;
    fM01  : LREAL;
    fM20  : LREAL;
    fM11  : LREAL;
    fM02  : LREAL;
    fM30  : LREAL;
    fM21  : LREAL;
    fM12  : LREAL;
    fM03  : LREAL;
```

```
    fMu20 : LREAL;
    fMu11 : LREAL;
    fMu02 : LREAL;
    fMu30 : LREAL;
    fMu21 : LREAL;
    fMu12 : LREAL;
    fMu03 : LREAL;
    fNu20 : LREAL;
    fNu11 : LREAL;
    fNu02 : LREAL;
    fNu30 : LREAL;
    fNu21 : LREAL;
    fNu12 : LREAL;
    fNu03 : LREAL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
fM00	LREAL	Spatial moment 00
fM10	LREAL	Spatial moment 10
fM01	LREAL	Spatial moment 01
fM20	LREAL	Spatial moment 20
fM11	LREAL	Spatial moment 11
fM02	LREAL	Spatial moment 02
fM30	LREAL	Spatial moment 30
fM21	LREAL	Spatial moment 21
fM12	LREAL	Spatial moment 12
fM03	LREAL	Spatial moment 03
fMu20	LREAL	Central moment 20
fMu11	LREAL	Central moment 11
fMu02	LREAL	Central moment 02
fMu30	LREAL	Central moment 30
fMu21	LREAL	Central moment 21
fMu12	LREAL	Central moment 12
fMu03	LREAL	Central moment 03
fNu20	LREAL	Normalized central moment 20
fNu11	LREAL	Normalized central moment 11
fNu02	LREAL	Normalized central moment 02
fNu30	LREAL	Normalized central moment 30
fNu21	LREAL	Normalized central moment 21
fNu12	LREAL	Normalized central moment 12
fNu03	LREAL	Normalized central moment 03

6.1.2.5.14 TcVnParamsAGAST

Offers parameters for AGAST method.

Syntax

Definition:

```
TYPE TcVnParamsAGAST :
STRUCT
    nThreshold         : DINT;
    bNonMaxSuppression : BOOL;
    eType              : ETcVnKeypointDetectionTypeAGAST;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
nThreshold	DINT	10	Threshold for the intensity difference between the center pixel and its surrounding circle
bNonMaxSuppression	BOOL	TRUE	If true, non-maximum suppression is applied.
eType	ETcVnKeypointDetectionTypeAGAST [▶ 218]	TCVN_KDT_AGAST_9_16	Neighborhood type

6.1.2.5.15 TcVnParamsAKAZE

Offers parameters for AKAZE method.

Syntax

Definition:

```
TYPE TcVnParamsAKAZE :
STRUCT
    eDescrType     : ETcVnFeatureDescriptorTypeAKAZE;
    nDescrSize     : UDINT;
    nDescrChannels : UDINT;
    fThreshold     : REAL;
    nOctaves       : UDINT;
    nOctaveLayers  : UDINT;
    eDiffusivity   : ETcVnDiffusivityTypeKAZE;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
eDescrType	ETcVnFeatureDescriptorTypeAKAZE [▶ 212]	TCVN_FDT_AKAZE_MLDB	Type of the descriptor
nDescrSize	UDINT	0	Size of the descriptor in bits (only for MLDB; 0 = full size)
nDescrChannels	UDINT	3	Number of descriptor channels (currently only 3 supported for MLDB)
fThreshold	REAL	0.001	Detector response threshold
nOctaves	UDINT	2	Maximum octave evolution
nOctaveLayers	UDINT	1	Number of sublevels per scale level
eDiffusivity	ETcVnDiffusivityTypeKAZE [▶ 205]	TCVN_DT1_KAZE_PM_G2	Diffusivity type

6.1.2.5.16 TcVnParamsBlobDetection

Offers parameters for blob detection

Syntax

Definition:

```
TYPE TcVnParamsBlobDetection :
STRUCT
    bFilterByArea          : BOOL;
    bFilterByCircularity   : BOOL;
    bFilterByConvexity     : BOOL;
    bFilterByEccentricity  : BOOL;
    bFilterByInertiaRatio  : BOOL;
    fMinArea               : REAL;
    fMaxArea               : REAL;
    fMinCircularity        : REAL;
    fMaxCircularity        : REAL;
    fMinConvexity          : REAL;
    fMaxConvexity          : REAL;
    fMinEccentricity       : REAL;
    fMaxEccentricity       : REAL;
    fMinInertiaRatio       : REAL;
    fMaxInertiaRatio       : REAL;
    eThresholdType         : ETcVnThresholdType;
    fMinThreshold          : REAL;
    fMaxThreshold          : REAL;
    fThresholdStep         : REAL;
    fMinBlobDistance       : REAL;
    nMinRepeatability      : UDINT;
    eBlobCombination       : ETcVnBlobCombination;
    bExcludeBorderBlobs    : BOOL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
bFilterByArea	BOOL	TRUE	Enable filtering by area (fMinArea <= area <= fMaxArea); strongly recommended to activate for filtering noise with fMinArea.
bFilterByCircularity	BOOL	FALSE	Enable filtering by circularity (fMinCircularity <= circularity(4pi * area / perimeter^2) <= fMaxCircularity).
bFilterByConvexity	BOOL	FALSE	Enable filtering by convexity (fMinConvexity <= convexity(area / hullArea) <= fMaxConvexity).
bFilterByEccentricity	BOOL	FALSE	Enable filtering by eccentricity (fMinEccentricity <= eccentricity <= fMaxEccentricity).
bFilterByInertiaRatio	BOOL	FALSE	Enable filtering by inertia ratio (fMinInertiaRatio <= inertia ratio <= fMaxInertiaRatio).
fMinArea	REAL	10	Min estimated blob area in pixel
fMaxArea	REAL	100000000	Max estimated blob area in pixel
fMinCircularity	REAL	0	Min circularity (1.0: ideal circle, < 1: less circular, 0: not circular at all)
fMaxCircularity	REAL	1	Max circularity (1.0: ideal circle, < 1: less circular, 0: not circular at all)
fMinConvexity	REAL	0	Min convexity (1.0: blob fully convex, < 1: less convex)
fMaxConvexity	REAL	1	Max convexity (1.0: blob fully convex, < 1: less convex)
fMinEccentricity	REAL	0	Min eccentricity (0.0: circular, 1.0: linear)
fMaxEccentricity	REAL	1	Max eccentricity (0.0: circular, 1.0: linear)
fMinInertiaRatio	REAL	0	Min inertia ratio (1.0: equal width and height, 0.0: linear)
fMaxInertiaRatio	REAL	1	Max inertia ratio (1.0: equal width and height, 0.0: linear)
eThresholdType	ETcVnThresholdType [▶ 232]	TCVN_TT_BINARY	Threshold type for internally applied threshold method (OTSU_XXX only supported for USINT images).

Name	Type	Default	Description
fMinThreshold	REAL	30	Threshold to start with (if fThresholdStep > 0, otherwise this is the only threshold used).
fMaxThreshold	REAL	225	Threshold to end with (if fThresholdStep > 0, otherwise this value is not used).
fThresholdStep	REAL	0	Sets to 0 if only 1 threshold should be used (much faster than multiple thresholds and combining the results).
fMinBlobDistance	REAL	5	Minimum distance between the center points of two different blobs (only used if fThresholdStep > 0; if distance < fMinBlobDistance, the blobs are treated as the same).
nMinRepeatability	UDINT	2	Minimum number of threshold steps, for which the same contour has to be detected (only used if fThresholdStep > 0; same means center point distance < fMinBlobDistance).
eBlobCombination	ETcVnBlobCombination [▶ 176]	TCVN_BC_MEDIAN_THRESHOLD	Selects, which of the multi-threshold blob contours should be returned.
bExcludeBorderBlobs	BOOL	FALSE	If true, blobs touching the image border are excluded from the results.

6.1.2.5.17 TcVnParamsBRISK

Offers parameters for BRISK method.

Syntax

Definition:

```
TYPE TcVnParamsBRISK :
STRUCT
    nThreshold    : DINT;
    nOctaves      : UDINT;
    fPatternScale : REAL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
nThreshold	DINT	30	Detection threshold
nOctaves	UDINT	3	Detection octaves (0 for single scale)
fPatternScale	REAL	1	Scale factor for the neighborhood pattern

6.1.2.5.18 TcVnParamsFAST

Offers parameters for FAST method.

Syntax

Definition:

```
TYPE TcVnParamsFAST :
STRUCT
    nThreshold         : DINT;
    bNonMaxSuppression : BOOL;
    eType              : ETcVnKeypointDetectionTypeFAST;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
nThreshold	DINT	10	Threshold for the intensity difference between the center pixel and its surrounding circle
bNonMaxSuppression	BOOL	TRUE	If true, non-maximum suppression is applied.
eType	ETcVnKeypointDetectionTypeFAST [▶ 218]	TCVN_KDT_FAST_9_16	Neighborhood type

6.1.2.5.19 TcVnParamsGFTT

Offers parameters for GFTT method.

Syntax

Definition:

```
TYPE TcVnParamsGFTT :
STRUCT
    nMaxCorners        : UDINT;
    fQualityLevel      : LREAL;
    fMinDistance       : LREAL;
    nBlockSize         : UDINT;
    bUseHarrisDetector : BOOL;
    fHarrisK           : LREAL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
nMaxCorners	UDINT	1000	Maximum number of corners to return (strongest ones)
fQualityLevel	LREAL	0.01	Minimum accepted corner quality, relative to the strongest one
fMinDistance	LREAL	1	Minimum euclidean distance between returned keypoints
nBlockSize	UDINT	3	Neighborhood block size
bUseHarrisDetector	BOOL	FALSE	If true, a Harris detector is used instead of the default method.
fHarrisK	LREAL	0.04	Free parameter of the Harris detector

6.1.2.5.20 TcVnParamsKAZE

Offers parameters for KAZE method

Syntax

Definition:

```
TYPE TcVnParamsKAZE :
STRUCT
    bExtended     : BOOL;
    bUpright      : BOOL;
    fThreshold    : REAL;
    nOctaves      : UDINT;
    nOctaveLayers : UDINT;
    eDiffusivity  : ETcVnDiffusivityTypeKAZE;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
bExtended	BOOL	FALSE	If true, the descriptor size is extended from the default 64 byte to 128 byte.
bUpright	BOOL	FALSE	If true, upright (rotation dependent) descriptors are computed.
fThreshold	REAL	0.001	Detector response threshold
nOctaves	UDINT	4	Maximum octave evolution of the image
nOctaveLayers	UDINT	2	Number of sublevels per scale level
eDiffusivity	ETcVnDiffusivityTypeKAZE [▶ 205]	TCVN_DT1_KAZE_PM_G2	Diffusivity type

6.1.2.5.21 TcVnParamsMSER

Offers parameters for MSER method

Syntax

Definition:

```
TYPE TcVnParamsMSER :
STRUCT
    nDelta         : DINT;
    nMinArea       : UDINT;
    nMaxArea       : UDINT;
    fMaxVariation  : LREAL;
    fMinDiversity  : LREAL;
    nMaxEvolution  : UDINT;
    fAreaThreshold : LREAL;
    fMinMargin     : LREAL;
    nEdgeBlurSize  : UDINT;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
nDelta	DINT	5	Delta for size comparison
nMinArea	UDINT	60	Minimum blob size
nMaxArea	UDINT	14400	Maximum blob size
fMaxVariation	LREAL	0.25	Maximum blob size variation
fMinDiversity	LREAL	0.2	Minimum MSER diversity
nMaxEvolution	UDINT	200	Maximum evolution steps (only used for color images)
fAreaThreshold	LREAL	1.01	Area threshold to cause re-initialization (only used for color images)
fMinMargin	LREAL	0.003	Minimum margin (only used for color images)
nEdgeBlurSize	UDINT	5	Aperture size for edge bluring (only used for color images)

6.1.2.5.22 TcVnParamsORB

Offers parameters for ORB method.

Syntax

Definition:

```
TYPE TcVnParamsORB :
STRUCT
    nMaxPoints     : UDINT;
    fPyramidScale  : REAL;
    nPyramidLevels : UDINT;
    nEdgeThreshold : UDINT;
    nFirstLevel    : UDINT;
    nBriefPoints   : UDINT;
    eScoreType     : ETcVnFeatureScoreTypeORB;
    nPatchSize     : UDINT;
    nFastThreshold : DINT;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
nMaxPoints	UDINT	500	Maximum number of returned keypoints
fPyramidScale	REAL	1.2	Pyramid decimation ratio (must be greater than 1, should be smaller than 2)
nPyramidLevels	UDINT	8	Number of pyramid levels
nEdgeThreshold	UDINT	31	Size of the border, where no features are detected (should match nPatchSize)
nFirstLevel	UDINT	0	First pyramid level (currently, only 0 is supported)
nBriefPoints	UDINT	2	Number of points to produce each BRIEF descriptor element (2, 3, 4)
eScoreType	ETcVnFeatureScoreTypeORB [▶ 213]	TCVN_FST_ORB_HARRIS	Score type (HARRIS is more stable but slightly slower than FAST)
nPatchSize	UDINT	31	Patch size of the BRIEF descriptor
nFastThreshold	DINT	20	Threshold for the FAST keypoint detection

6.1.2.5.23 TcVnParamsSB

Offers parameters for SB method (a simple blob detector with multiple thresholds).

Syntax

Definition:

```
TYPE TcVnParamsSB :
STRUCT
    bFilterByArea        : BOOL;
    bFilterByCircularity : BOOL;
    bFilterByColor       : BOOL;
    bFilterByConvexity   : BOOL;
    bFilterByInertia     : BOOL;
    fMinArea             : REAL;
    fMaxArea             : REAL;
    fMinCircularity      : REAL;
    fMaxCircularity      : REAL;
    nBlobColor           : USINT;
    fMinConvexity        : REAL;
    fMaxConvexity        : REAL;
    fMinInertiaRatio     : REAL;
    fMaxInertiaRatio     : REAL;
    fMinBlobDist         : REAL;
    nMinRepeatability    : UDINT;
    fMinThreshold        : REAL;
    fMaxThreshold        : REAL;
    fThresholdStep       : REAL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Default	Description
bFilterByArea	BOOL	TRUE	Enable filtering by area (fMinArea <= area < fMaxArea).
bFilterByCircularity	BOOL	FALSE	Enable filtering by circularity (fMinCircularity <= circularity(4pi * area / perimeter^2) < fMaxCircularity).
bFilterByColor	BOOL	FALSE	Enable filtering by color (thresholdedColor(0 or 255) = nBlobColor).
bFilterByConvexity	BOOL	FALSE	Enable filtering by convexity (fMinConvexity <= convexity(area / hullArea) < fMaxConvexity).
bFilterByInertia	BOOL	FALSE	Enable filtering by inertia ratio (fMinInertiaRatio <= inertia ratio < fMaxInertiaRatio).
fMinArea	REAL	25	Min estimated blob area in pixel
fMaxArea	REAL	15000	Max estimated blob area in pixel
fMinCircularity	REAL	0	Min circularity (1.0: ideal circle, < 1: less circular, 0: not circular at all)
fMaxCircularity	REAL	1	Max circularity (1.0: ideal circle, < 1: less circular, 0: not circular at all)
nBlobColor	USINT	255	0 or 255
fMinConvexity	REAL	0	Min convexity (1.0: blob fully convex, < 1: less convex)
fMaxConvexity	REAL	1	Max convexity (1.0: blob fully convex, < 1: less convex)
fMinInertiaRatio	REAL	0	Min inertia ratio (0.0 .. 1.0)
fMaxInertiaRatio	REAL	1	Max inertia ratio (0.0 .. 1.0)
fMinBlobDist	REAL	5	Min distance between different blobs
nMinRepeatability	UDINT	2	Min number of same detected blobs (dist < fMinBlobDist) in different thresholds
fMinThreshold	REAL	30	Min threshold (start)
fMaxThreshold	REAL	225	Max threshold (stop)
fThresholdStep	REAL	10	Threshold step between min and max threshold

6.1.2.5.24 TcVnPixelFormat

Contains detailed information about the pixel format.

Syntax

Definition:

```
TYPE TcVnPixelFormat :
STRUCT
    bSupported     : BIT;
    bSigned        : BIT;
    bPlanar        : BIT;
    bFloat         : BIT;
    nChannels      : BYTE;
    ePixelEncoding : ETcVnPixelEncoding;
    ePixelPackMode : ETcVnPixelPackMode;
    nElementSize   : UINT;
    nTotalSize     : UINT;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
bSupported	BIT	If false, the pixel format is not supported.
bSigned	BIT	If true, pixel intensities are signed values.
bPlanar	BIT	If true, the image channels are stored planar instead of interleaved (e.g. RRRRRGGGGGBBBBB instead of RGBRGBRGBRGBRGB).
bFloat	BIT	If true, the pixel format is floating point.
nChannels	BYTE	Number of channels
ePixelEncoding	ETcVnPixelEncoding [▶ 225]	Pixel encoding
ePixelPackMode	ETcVnPixelPackMode [▶ 225]	Pixel pack mode
nElementSize	UINT	Size (bit) of a single pixel channel
nTotalSize	UINT	Size (bit) of all pixel channels

6.1.2.5.25 TcVnRectangle_DINT

Contains origin and size of a rectangle.

Syntax

Definition:

```
TYPE TcVnRectangle_DINT :
STRUCT
    nX      : DINT;
    nY      : DINT;
    nWidth  : DINT;
    nHeight : DINT;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
nX	DINT	X coordinate of the top-left corner
nY	DINT	Y coordinate of the top-left corner
nWidth	DINT	Width
nHeight	DINT	Height

6.1.2.5.26 TcVnRectangle_UDINT

Contains origin and size of a rectangle.

Syntax

Definition:

```
TYPE TcVnRectangle_UDINT :
STRUCT
    nX      : UDINT;
    nY      : UDINT;
    nWidth  : UDINT;
    nHeight : UDINT;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
nX	UDINT	X coordinate of the top-left corner
nY	UDINT	Y coordinate of the top-left corner
nWidth	UDINT	Width
nHeight	UDINT	Height

6.1.2.5.27 TcVnRotatedRectangle

Contains center, size and angle of a rotated rectangle.

Syntax

Definition:

```
TYPE TcVnRotatedRectangle :
STRUCT
    aCenter : TcVnPoint2_REAL;
    stSize  : TcVnSize2_REAL;
    fAngle  : REAL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
aCenter	TcVnPoint2_REAL [▶ 171]	Center point
stSize	TcVnSize2_REAL [▶ 258]	Size composed of fWidth and fHeight
fAngle	REAL	Angle in degree

6.1.2.5.28 TcVnSize2_REAL

Contains width and height.

Syntax

Definition:

```
TYPE TcVnSize2_REAL :
STRUCT
    fWidth  : REAL;
    fHeight : REAL;
END_STRUCT
END_TYPE
```

Parameters

Name	Type	Description
fWidth	REAL	Width
fHeight	REAL	Height

6.1.3 Interfaces

Interfaces erfordern keine Lizenz.

Gruppierung von Interfaces

Alle Interfaces der TwinCAT Vision Bibliothek sind thematisch in die folgenden Gruppen sortiert:

- Bilder [▶ 391] ()
- Container [▶ 259] ()
- Machine Learning [▶ 433] ()
- Sonstiges [▶ 437] ()

6.1.3.1 Containers

Diese Gruppe enthält Interfaces zur Handhabung von Containern [▶ 163].

Generelle Informationen

Ein Container ist grundsätzlich ein Vektor von mehreren Elementen. Dabei kann die Anzahl der Elemente dynamisch verändert werden. Dafür wird automatisch entsprechender Speicher im Router-Speicher [▶ 56] allokiert und freigegeben.

Container-Typen

Alle Elemente eines Containers haben denselben Typ. Auf Basis des Typs der enthaltenen Elemente werden Container durch eine eindeutige GUID charakterisiert. Zur einfacheren Handhabung sind diese GUIDs durch Konstanten [▶ 171] definiert. Alle Typ-Bezeichnungen für Container beginnen mit „ContainerType_Vector_“ (mit der Ausnahme „ContainerType_String_SINT“). Für alle API-Elemente, die Gebrauch von Containern machen, sind Typ-Anforderungen ggfs. dokumentiert.

Der Typ eines Containers kann folgendermaßen in Erfahrung gebracht werden:

```
ipContainer.GetElementTypeGuid(nTypeGuid);
```

6.1.3.1.1 Function Blocks

Diese Gruppe enthält Interfaces zur Implementierung bestimmter Funktionsblöcke.

6.1.3.1.1.1 ITcVnCustomContainerOperation_ITcVnContainer

Offers an interface for custom, elementwise container operations (for containers of containers) with up to 3 different containers.

Inheritance Hierarchy

ITcVnCustomContainerOperation_ITcVnContainer

Methods

Name	Origin	Description
Execute [▶ 260]	ITcVnCustomContainerOperation_ITcVnContainer	Executes the custom operation on the container elements.

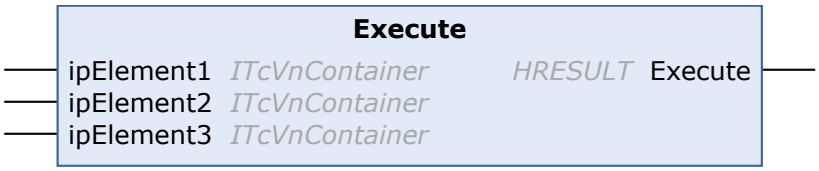
Verwandte Funktionen

- F_VN_CustomElementWiseContainerOperation_ITcVnContainer [▶ 1004]

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.1.1.1 Execute



Executes the custom operation on the container elements.

Syntax

Definition:

```
METHOD Execute : HRESULT
VAR_INPUT
    ipElement1 : ITcVnContainer;
    ipElement2 : ITcVnContainer;
    ipElement3 : ITcVnContainer;
END_VAR
```

Inputs

Name	Type	Description
ipElement1	ITcVnContainer [▶ 385]	Current element of the 1st container.
ipElement2	ITcVnContainer [▶ 385]	Current element of the 2nd container.
ipElement3	ITcVnContainer [▶ 385]	Current element of the 3rd container.

Return value

HRESULT [▶ 150]

6.1.3.1.1.2 ITcVnCustomContainerOperation_ITcVnForwardIterator

Offers an interface for custom, elementwise container operations with up to 3 different containers.

Inheritance Hierarchy

ITcVnCustomContainerOperation_ITcVnForwardIterator

Methods

Name	Origin	Description
Execute [▶ 261]	ITcVnCustomContainerOperation_ITcVnForwardIterator	Executes the custom operation on the container elements.

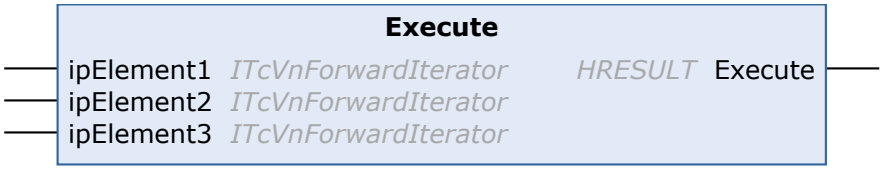
Verwandte Funktionen

- F_VN_CustomElementWiseContainerOperation_ITcVnForwardIterator [▶ 1005]

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.1.2.1 Execute



Executes the custom operation on the container elements.

Syntax

Definition:

```
METHOD Execute : HRESULT
VAR_INPUT
    ipElement1 : ITcVnForwardIterator;
    ipElement2 : ITcVnForwardIterator;
    ipElement3 : ITcVnForwardIterator;
END_VAR
```

Inputs

Name	Type	Description
ipElement1	ITcVnForwardIterator [▶ 376]	Current element of the 1st container.
ipElement2	ITcVnForwardIterator [▶ 376]	Current element of the 2nd container.
ipElement3	ITcVnForwardIterator [▶ 376]	Current element of the 3rd container.

Return value

HRESULT [▶ 150]

6.1.3.1.1.3 ITcVnCustomElementCondition_ITcVnContainer

Offers an interface for a custom condition computed for a container.

Inheritance Hierarchy

ITcVnCustomElementCondition_ITcVnContainer

Methods

Name	Origin	Description
Condition [▶ 262]	ITcVnCustomElementCondition_ITcVnContainer	Evaluates the condition for the container.

Verwandte Funktionen

- F_VN_CopyContainerElementsConditional_ITcVnContainer [▶ 773]

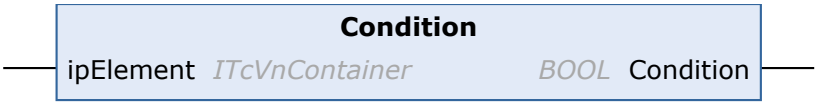
Beispiele

- Selektieren von Container-Elementen [▶ 2840]

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.1.3.1 Condition



Evaluates the condition for the container.

Syntax

Definition:

```
METHOD Condition : BOOL
VAR_INPUT
    ipElement : ITcVnContainer;
END_VAR
```

Inputs

Name	Type	Description
ipElement	ITcVnContainer [▶ 385]	The container for which the condition is evaluated.

Return value

BOOL

6.1.3.1.1.4 ITcVnCustomElementCondition_ITcVnForwardIterator

Offers an interface for a custom condition computed for an element represented by an ITcVnForwardIterator.

Inheritance Hierarchy

ITcVnCustomElementCondition_ITcVnForwardIterator

Methods

Name	Origin	Description
Condition [▶ 263]	ITcVnCustomElementCondition_ITcVnForwardIterator	Evaluates the condition for the provided element.

Verwandte Funktionen

- F_VN_CopyContainerElementsConditional_ITcVnForwardIterator [▶ 774]

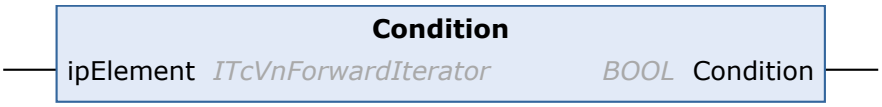
Beispiele

- Selektieren von Container-Elementen [▶ 2840]

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.1.4.1 Condition



Evaluates the condition for the provided element.

Syntax

Definition:

```
METHOD Condition : BOOL
VAR_INPUT
    ipElement : ITcVnForwardIterator;
END_VAR
```

Inputs

Name	Type	Description
ipElement	ITcVnForwardIterator [▶ 376]	Element for which the condition is evaluated.

Return value

BOOL

6.1.3.1.2 ITcVnAccess

Das Interface `ITcVnAccess` dient dem Zugriff auf Elemente eines Containers [▶ 164] und wird von einem Iterator [▶ 375] bereitgestellt. Im Gegensatz zu ITcVnRandomAccess [▶ 315] kann nur das Element adressiert werden, auf das der zugehörige Iterator in dem Moment zeigt.

6.1.3.1.2.1 ITcVnAccess_DINT

Offers an access interface for DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 264]	ITcVnAccess_DINT	Gets the value
Set [▶ 264]	ITcVnAccess_DINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.1.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    nValue : Reference To DINT;
END_VAR
```

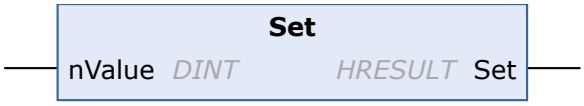
Inputs

Name	Type	Description
nValue	Reference To DINT	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.1.2 Set



Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    nValue : DINT;
END_VAR
```

Inputs

Name	Type	Description
nValue	DINT	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.2 ITcVnAccess_INT

Offers an access interface for INT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_INT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 265]	ITcVnAccess_INT	Gets the value
Set [▶ 266]	ITcVnAccess_INT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.2.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    nValue : Reference To INT;
END_VAR
```

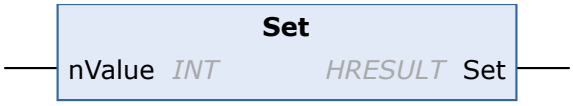
Inputs

Name	Type	Description
nValue	Reference To INT	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.2.2 Set



Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    nValue : INT;
END_VAR
```

Inputs

Name	Type	Description
nValue	INT	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.3 ITcVnAccess_ITcVnImage

Offers an access interface for images.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_ITcVnImage

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 267]	ITcVnAccess_ITcVnImage	Gets the image.
Set [▶ 267]	ITcVnAccess_ITcVnImage	Sets the image.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.3.1 Get

Gets the image.

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    ipImage : Reference To ITcVnImage;
END_VAR
```

Inputs

Name	Type	Description
ipImage	Reference To ITcVnImage [▶ 426]	Returns the image.

Return value

HRESULT [▶ 150]

6.1.3.1.2.3.2 Set

Sets the image.

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    ipImage : ITcVnImage;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	The image to set.

Return value

HRESULT [▶ 150]

6.1.3.1.2.4 ITcVnAccess_LREAL

Offers an access interface for LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 268]	ITcVnAccess_LREAL	Gets the value
Set [▶ 268]	ITcVnAccess_LREAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.4.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    fValue : Reference To LREAL;
END_VAR
```

Inputs

Name	Type	Description
fValue	Reference To LREAL	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.4.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    fValue : LREAL;
END_VAR
```

Inputs

Name	Type	Description
fValue	LREAL	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.5 ITcVnAccess_REAL

Offers an access interface for REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 269]	ITcVnAccess_REAL	Gets the value
Set [▶ 270]	ITcVnAccess_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.5.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    fValue : Reference To REAL;
END_VAR
```

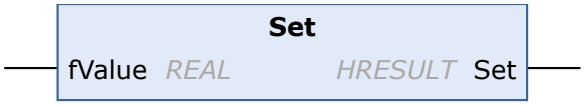
Inputs

Name	Type	Description
fValue	Reference To REAL	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.5.2 Set



Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    fValue : REAL;
END_VAR
```

Inputs

Name	Type	Description
fValue	REAL	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.6 ITcVnAccess_SINT

Offers an access interface for SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 271]	ITcVnAccess_SINT	Gets the value
Set [▶ 271]	ITcVnAccess_SINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.6.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    nValue : Reference To SINT;
END_VAR
```

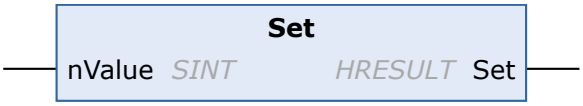
Inputs

Name	Type	Description
nValue	Reference To SINT	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.6.2 Set



Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    nValue : SINT;
END_VAR
```

Inputs

Name	Type	Description
nValue	SINT	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.7 ITcVnAccess_TcVnDMatch

Offers an access interface for TcVnDMatch values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnDMatch

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 272]	ITcVnAccess_TcVnDMatch	Gets the value
Set [▶ 272]	ITcVnAccess_TcVnDMatch	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.7.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    stDMatch : Reference To TcVnDMatch;
END_VAR
```

Inputs

Name	Type	Description
stDMatch	Reference To TcVnDMatch [▶ 244]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.7.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    stDMatch : Reference To TcVnDMatch;
END_VAR
```

Inputs

Name	Type	Description
stDMatch	Reference To TcVnDMatch [▶ 244]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.8 ITcVnAccess_TcVnKeyPoint

Offers an access interface for TcVnKeyPoint values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnKeyPoint

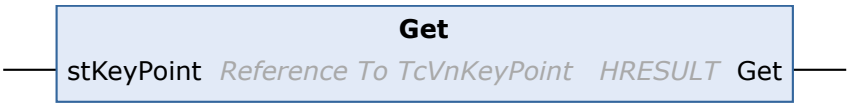
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 273]	ITcVnAccess_TcVnKeyPoint	Gets the value.
Set [▶ 274]	ITcVnAccess_TcVnKeyPoint	Sets the value.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.8.1 Get



Gets the value.

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    stKeyPoint : Reference To TcVnKeyPoint;
END_VAR
```

Inputs

Name	Type	Description
stKeyPoint	Reference To TcVnKeyPoint [▶ 244]	Returns the value.

Return value

HRESULT [▶ 150]

6.1.3.1.2.8.2 Set

Sets the value.

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    stKeyPoint : Reference To TcVnKeyPoint;
END_VAR
```

Inputs

Name	Type	Description
stKeyPoint	Reference To TcVnKeyPoint [▶ 244]	The value to set.

Return value

HRESULT [▶ 150]

6.1.3.1.2.9 ITcVnAccess_TcVnPoint2_DINT

Offers an access interface for TcVnPoint2_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnPoint2_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 275]	ITcVnAccess_TcVnPoint2_DINT	Gets the value.
Set [▶ 275]	ITcVnAccess_TcVnPoint2_DINT	Sets the value.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.9.1 Get

Gets the value.

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aPoint : Reference To TcVnPoint2_DINT;
END_VAR
```

Inputs

Name	Type	Description
aPoint	Reference To TcVnPoint2_DINT [▶ 171]	Returns the value.

Return value

HRESULT [▶ 150]

6.1.3.1.2.9.2 Set

Sets the value.

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aPoint : Reference To TcVnPoint2_DINT;
END_VAR
```

Inputs

Name	Type	Description
aPoint	Reference To TcVnPoint2_DINT [▶ 171]	The value to set.

Return value

HRESULT [▶ 150]

6.1.3.1.2.10 ITcVnAccess_TcVnPoint2_LREAL

Offers an access interface for TcVnPoint2_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnPoint2_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 276]	ITcVnAccess_TcVnPoint2_LREAL	Gets the value
Set [▶ 277]	ITcVnAccess_TcVnPoint2_LREAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.10.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aPoint : Reference To TcVnPoint2_LREAL;
END_VAR
```

Inputs

Name	Type	Description
aPoint	Reference To TcVnPoint2_LREAL [▶ 171]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.10.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aPoint : Reference To TcVnPoint2_LREAL;
END_VAR
```

Inputs

Name	Type	Description
aPoint	Reference To TcVnPoint2_LREAL [▶ 171]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.11 ITcVnAccess_TcVnPoint2_REAL

Offers an access interface for TcVnPoint2_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnPoint2_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 278]	ITcVnAccess_TcVnPoint2_REAL	Gets the value
Set [▶ 278]	ITcVnAccess_TcVnPoint2_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.11.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aPoint : Reference To TcVnPoint2_REAL;
END_VAR
```

Inputs

Name	Type	Description
aPoint	Reference To TcVnPoint2_REAL [▶ 171]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.11.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aPoint : Reference To TcVnPoint2_REAL;
END_VAR
```

Inputs

Name	Type	Description
aPoint	Reference To TcVnPoint2_REAL [▶ 171]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.12 ITcVnAccess_TcVnPoint3_LREAL

Offers an access interface for TcVnPoint3_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnPoint3_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 279]	ITcVnAccess_TcVnPoint3_LREAL	Gets the value
Set [▶ 279]	ITcVnAccess_TcVnPoint3_LREAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.12.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aPoint : Reference To TcVnPoint3_LREAL;
END_VAR
```

Inputs

Name	Type	Description
aPoint	Reference To TcVnPoint3_LREAL [▶ 171]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.12.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aPoint : Reference To TcVnPoint3_LREAL;
END_VAR
```

Inputs

Name	Type	Description
aPoint	Reference To TcVnPoint3_LREAL [▶ 171]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.13 ITcVnAccess_TcVnPoint3_REAL

Offers an access interface for TcVnPoint3_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnPoint3_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 280]	ITcVnAccess_TcVnPoint3_REAL	Gets the value
Set [▶ 281]	ITcVnAccess_TcVnPoint3_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.13.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aPoint : Reference To TcVnPoint3_REAL;
END_VAR
```

Inputs

Name	Type	Description
aPoint	Reference To TcVnPoint3_REAL [▶ 171]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.13.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aPoint : Reference To TcVnPoint3_REAL;
END_VAR
```

Inputs

Name	Type	Description
aPoint	Reference To TcVnPoint3_REAL [▶ 171]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.14 ITcVnAccess_TcVnRectangle_DINT

Offers an access interface for TcVnRectangle_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnRectangle_DINT

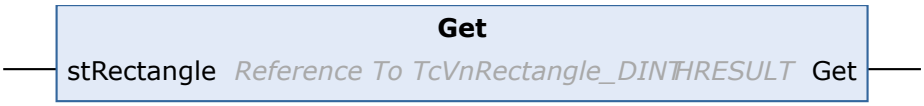
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 282]	ITcVnAccess_TcVnRectangle_DINT	Gets the value
Set [▶ 282]	ITcVnAccess_TcVnRectangle_DINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.14.1 Get



Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    stRectangle : Reference To TcVnRectangle_DINT;
END_VAR
```

Inputs

Name	Type	Description
stRectangle	Reference To TcVnRectangle_DINT [▶ 257]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.14.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    stRectangle : Reference To TcVnRectangle_DINT;
END_VAR
```

Inputs

Name	Type	Description
stRectangle	Reference To TcVnRectangle_DINT [▶ 257]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.15 ITcVnAccess_TcVnVector2_DINT

Offers an access interface for TcVnVector2_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector2_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 283]	ITcVnAccess_TcVnVector2_DINT	Gets the value
Set [▶ 284]	ITcVnAccess_TcVnVector2_DINT	Sets the value.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.15.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector2_DINT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector2_DINT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.15.2 Set

Sets the value.

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector2_DINT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector2_DINT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.16 ITcVnAccess_TcVnVector2_INT

Offers an access interface for TcVnVector2_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector2_INT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 285]	ITcVnAccess_TcVnVector2_INT	Gets the value
Set [▶ 285]	ITcVnAccess_TcVnVector2_INT	Sets the value.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.16.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector2_INT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector2_INT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.16.2 Set

Sets the value.

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector2_INT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector2_INT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.17 ITcVnAccess_TcVnVector2_REAL

Offers an access interface for TcVnVector2_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector2_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 286]	ITcVnAccess_TcVnVector2_REAL	Gets the value
Set [▶ 286]	ITcVnAccess_TcVnVector2_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.17.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector2_REAL;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector2_REAL [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.17.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector2_REAL;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector2_REAL [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.18 ITcVnAccess_TcVnVector2_SINT

Offers an access interface for TcVnVector2_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector2_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 287]	ITcVnAccess_TcVnVector2_SINT	Gets the value
Set [▶ 288]	ITcVnAccess_TcVnVector2_SINT	Sets the value.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.18.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector2_SINT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector2_SINT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.18.2 Set

Sets the value.

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector2_SINT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector2_SINT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.19 ITcVnAccess_TcVnVector2_UINT

Offers an access interface for TcVnVector2_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector2_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 289]	ITcVnAccess_TcVnVector2_UINT	Gets the value
Set [▶ 289]	ITcVnAccess_TcVnVector2_UINT	Sets the value.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.19.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector2_UINT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector2_UINT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.19.2 Set

Sets the value.

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector2_UINT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector2_UINT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.20 ITcVnAccess_TcVnVector2_USINT

Offers an access interface for TcVnVector2_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector2_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 290]	ITcVnAccess_TcVnVector2_USINT	Gets the value
Set [▶ 291]	ITcVnAccess_TcVnVector2_USINT	Sets the value.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.20.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector2_USINT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector2_USINT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.20.2 Set

Sets the value.

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector2_USINT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector2_USINT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.21 ITcVnAccess_TcVnVector3_INT

Offers an access interface for TcVnVector3_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector3_INT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 292]	ITcVnAccess_TcVnVector3_INT	Gets the value
Set [▶ 292]	ITcVnAccess_TcVnVector3_INT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.21.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector3_INT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector3_INT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.21.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector3_INT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector3_INT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.22 ITcVnAccess_TcVnVector3_REAL

Offers an access interface for TcVnVector3_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector3_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 293]	ITcVnAccess_TcVnVector3_REAL	Gets the value
Set [▶ 294]	ITcVnAccess_TcVnVector3_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.22.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector3_REAL;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector3_REAL [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.22.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector3_REAL;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector3_REAL [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.23 ITcVnAccess_TcVnVector3_SINT

Offers an access interface for TcVnVector3_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector3_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 295]	ITcVnAccess_TcVnVector3_SINT	Gets the value
Set [▶ 295]	ITcVnAccess_TcVnVector3_SINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.23.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector3_SINT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector3_SINT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.23.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector3_SINT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector3_SINT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.24 ITcVnAccess_TcVnVector3_UINT

Offers an access interface for TcVnVector3_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector3_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 296]	ITcVnAccess_TcVnVector3_UINT	Gets the value
Set [▶ 296]	ITcVnAccess_TcVnVector3_UINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.24.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector3_UINT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector3_UINT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.24.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector3_UINT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector3_UINT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.25 ITcVnAccess_TcVnVector3_USINT

Offers an access interface for TcVnVector3_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector3_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 297]	ITcVnAccess_TcVnVector3_USINT	Gets the value
Set [▶ 298]	ITcVnAccess_TcVnVector3_USINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.25.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector3_USINT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector3_USINT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.25.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector3_USINT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector3_USINT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.26 ITcVnAccess_TcVnVector4_DINT

Offers an access interface for TcVnVector4_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector4_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 299]	ITcVnAccess_TcVnVector4_DINT	Gets the value
Set [▶ 299]	ITcVnAccess_TcVnVector4_DINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.26.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector4_DINT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector4_DINT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.26.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector4_DINT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector4_DINT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.27 ITcVnAccess_TcVnVector4_INT

Offers an access interface for TcVnVector4_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector4_INT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 300]	ITcVnAccess_TcVnVector4_INT	Gets the value
Set [▶ 301]	ITcVnAccess_TcVnVector4_INT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.27.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector4_INT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector4_INT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.27.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector4_INT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector4_INT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.28 ITcVnAccess_TcVnVector4_LREAL

Offers an access interface for TcVnVector4_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector4_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 302]	ITcVnAccess_TcVnVector4_LREAL	Gets the value
Set [▶ 302]	ITcVnAccess_TcVnVector4_LREAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.28.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector4_LREAL;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector4_LREAL [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.28.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector4_LREAL;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector4_LREAL [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.29 ITcVnAccess_TcVnVector4_SINT

Offers an access interface for TcVnVector4_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector4_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 303]	ITcVnAccess_TcVnVector4_SINT	Gets the value
Set [▶ 304]	ITcVnAccess_TcVnVector4_SINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.29.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector4_SINT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector4_SINT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.29.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector4_SINT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector4_SINT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.30 ITcVnAccess_TcVnVector4_UINT

Offers an access interface for TcVnVector4_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector4_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 304]	ITcVnAccess_TcVnVector4_UINT	Gets the value
Set [▶ 305]	ITcVnAccess_TcVnVector4_UINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.30.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector4_UINT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector4_UINT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.30.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector4_UINT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector4_UINT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.31 ITcVnAccess_TcVnVector4_USINT

Offers an access interface for TcVnVector4_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector4_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 306]	ITcVnAccess_TcVnVector4_USINT	Gets the value
Set [▶ 306]	ITcVnAccess_TcVnVector4_USINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.31.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector4_USINT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector4_USINT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.31.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector4_USINT;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector4_USINT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.32 ITcVnAccess_UDINT

Offers an access interface for UDINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_UDINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 307]	ITcVnAccess_UDINT	Gets the value
Set [▶ 308]	ITcVnAccess_UDINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.32.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    nValue : Reference To UDINT;
END_VAR
```

Inputs

Name	Type	Description
nValue	Reference To UDINT	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.32.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    nValue : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nValue	UDINT	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.33 ITcVnAccess_UINT

Offers an access interface for UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 309]	ITcVnAccess_UINT	Gets the value
Set [▶ 309]	ITcVnAccess_UINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.33.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    nValue : Reference To UINT;
END_VAR
```

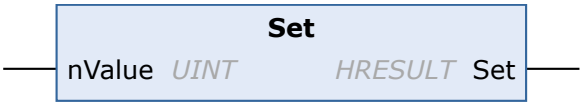
Inputs

Name	Type	Description
nValue	Reference To UINT	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.33.2 Set



Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    nValue : UINT;
END_VAR
```

Inputs

Name	Type	Description
nValue	UINT	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.34 ITcVnAccess_ULINT

Offers an access interface for ULINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_ULINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 310]	ITcVnAccess_ULINT	Gets the value
Set [▶ 310]	ITcVnAccess_ULINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.34.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    nValue : Reference To ULINT;
END_VAR
```

Inputs

Name	Type	Description
nValue	Reference To ULINT	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.34.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    nValue : ULINT;
END_VAR
```

Inputs

Name	Type	Description
nValue	ULINT	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.35 ITcVnAccess_USINT

Offers an access interface for USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 311]	ITcVnAccess_USINT	Gets the value
Set [▶ 312]	ITcVnAccess_USINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.35.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    nValue : Reference To USINT;
END_VAR
```

Inputs

Name	Type	Description
nValue	Reference To USINT	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.35.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    nValue : USINT;
END_VAR
```

Inputs

Name	Type	Description
nValue	USINT	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.36 ITcVnAccess_TcVnVector4_REAL

Offers an access interface for TcVnVector4_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector4_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 313]	ITcVnAccess_TcVnVector4_REAL	Gets the value
Set [▶ 313]	ITcVnAccess_TcVnVector4_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.36.1 Get

Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector4_REAL;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector4_REAL [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.36.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    aVector : Reference To TcVnVector4_REAL;
END_VAR
```

Inputs

Name	Type	Description
aVector	Reference To TcVnVector4_REAL [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.2.37 ITcVnAccess_TcVnRotatedRectangle

Offers an access interface for TcVnRotatedRectangle values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnRotatedRectangle

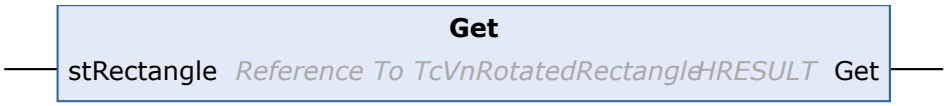
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 314]	ITcVnAccess_TcVnRotatedRectangle	Gets the value
Set [▶ 314]	ITcVnAccess_TcVnRotatedRectangle	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.2.37.1 Get



Gets the value

Syntax

Definition:

```
METHOD Get : HRESULT
VAR_INPUT
    stRectangle : Reference To TcVnRotatedRectangle;
END_VAR
```

Inputs

Name	Type	Description
stRectangle	Reference To TcVnRotatedRectangle [▶ 258]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.2.37.2 Set

Sets the value

Syntax

Definition:

```
METHOD Set : HRESULT
VAR_INPUT
    stRectangle : Reference To TcVnRotatedRectangle;
END_VAR
```

Inputs

Name	Type	Description
stRectangle	Reference To TcVnRotatedRectangle [▶ 258]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3 ITcVnRandomAccess

Das Interface `ITcVnRandomAccess` dient dem beliebigen Zugriff auf Elemente eines Containers [▶ 164] und wird von einem ITcVnRandomAccessIterator [▶ 382] bereitgestellt. Im Gegensatz zu ITcVnAccess [▶ 263] können beliebige Elemente eines Containers adressiert werden.

6.1.3.1.3.1 ITcVnRandomAccess_DINT

Offers a random access interface for DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_DINT [▶ 263]
    ITcVnRandomAccess_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 264]	ITcVnAccess_DINT	Gets the value
Set [▶ 264]	ITcVnAccess_DINT	Sets the value
GetAt [▶ 316]	ITcVnRandomAccess_DINT	Gets the value
SetAt [▶ 316]	ITcVnRandomAccess_DINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.1.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue  : Reference To DINT;
END_VAR
```

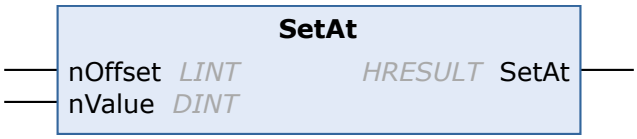
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
nValue	Reference To DINT	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.1.2 SetAt



Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue  : DINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
nValue	DINT	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.2 ITcVnRandomAccess_INT

Offers a random access interface for INT values

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_INT [▶ 265]
    ITcVnRandomAccess_INT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 265]	ITcVnAccess_INT	Gets the value
Set [▶ 266]	ITcVnAccess_INT	Sets the value
GetAt [▶ 317]	ITcVnRandomAccess_INT	Gets the value
SetAt [▶ 318]	ITcVnRandomAccess_INT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.2.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue  : Reference To INT;
END_VAR
```

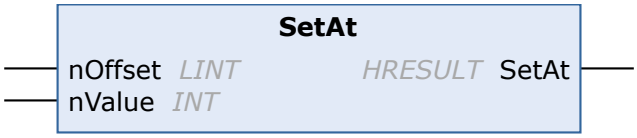
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
nValue	Reference To INT	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.2.2 SetAt



Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue  : INT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
nValue	INT	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.3 ITcVnRandomAccess_ITcVnImage

Offers an random access interface for images.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_ITcVnImage [▶ 266]
    ITcVnRandomAccess_ITcVnImage

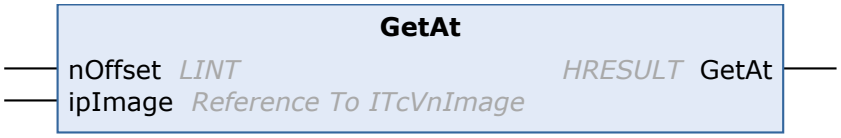
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 267]	ITcVnAccess_ITcVnImage	Gets the image.
Set [▶ 267]	ITcVnAccess_ITcVnImage	Sets the image.
GetAt [▶ 319]	ITcVnRandomAccess_ITcVnImage	Gets the image.
SetAt [▶ 319]	ITcVnRandomAccess_ITcVnImage	Sets the image.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.3.1 GetAt



Gets the image.

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    ipImage : Reference To ITcVnImage;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position.
ipImage	Reference To ITcVnImage [▶ 426]	Returns the image.

Return value

HRESULT [▶ 150]

6.1.3.1.3.3.2 SetAt

Sets the image.

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    ipImage : ITcVnImage;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
ipImage	ITcVnImage [▶ 426]	The image to set.

Return value

HRESULT [▶ 150]

6.1.3.1.3.4 ITcVnRandomAccess_LREAL

Offers a random access interface for LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_LREAL [▶ 268]
    ITcVnRandomAccess_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 268]	ITcVnAccess_LREAL	Gets the value
Set [▶ 268]	ITcVnAccess_LREAL	Sets the value
GetAt [▶ 320]	ITcVnRandomAccess_LREAL	Gets the value
SetAt [▶ 321]	ITcVnRandomAccess_LREAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.4.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    fValue  : Reference To LREAL;
END_VAR
```

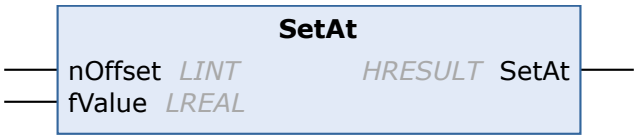
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
fValue	Reference To LREAL	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.4.2 SetAt



Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    fValue  : LREAL;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position.
fValue	LREAL	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.5 ITcVnRandomAccess_REAL

Offers a random access interface for REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_REAL [▶ 269]
    ITcVnRandomAccess_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 269]	ITcVnAccess_REAL	Gets the value
Set [▶ 270]	ITcVnAccess_REAL	Sets the value
GetAt [▶ 322]	ITcVnRandomAccess_REAL	Gets the value
SetAt [▶ 322]	ITcVnRandomAccess_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.5.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    fValue  : Reference To REAL;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
fValue	Reference To REAL	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.5.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    fValue  : REAL;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
fValue	REAL	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.6 ITcVnRandomAccess_SINT

Offers a random access interface for SINT values

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_SINT [▶ 270]
    ITcVnRandomAccess_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 271]	ITcVnAccess_SINT	Gets the value
Set [▶ 271]	ITcVnAccess_SINT	Sets the value
GetAt [▶ 323]	ITcVnRandomAccess_SINT	Gets the value
SetAt [▶ 324]	ITcVnRandomAccess_SINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.6.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue  : Reference To SINT;
END_VAR
```

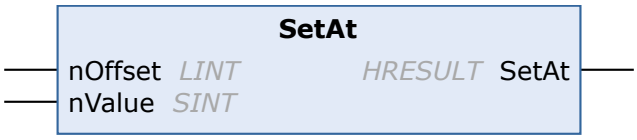
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
nValue	Reference To SINT	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.6.2 SetAt



Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue  : SINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
nValue	SINT	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.7 ITcVnRandomAccess_TcVnDMatch

Offers an random access interface for TcVnDMatch values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnDMatch [▶ 272]
    ITcVnRandomAccess_TcVnDMatch

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 272]	ITcVnAccess_TcVnDMatch	Gets the value
Set [▶ 272]	ITcVnAccess_TcVnDMatch	Sets the value
GetAt [▶ 325]	ITcVnRandomAccess_TcVnDMatch	Gets the value
SetAt [▶ 325]	ITcVnRandomAccess_TcVnDMatch	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.7.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset  : LINT;
    stDMatch : Reference To TcVnDMatch;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position.
stDMatch	Reference To TcVnDMatch [▶ 244]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.7.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset  : LINT;
    stDMatch : Reference To TcVnDMatch;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
stDMatch	Reference To TcVnDMatch [▶ 244]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.8 ITcVnRandomAccess_TcVnKeyPoint

Offers a random access interface for TcVnKeyPoint values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnKeyPoint [▶ 273]
    ITcVnRandomAccess_TcVnKeyPoint

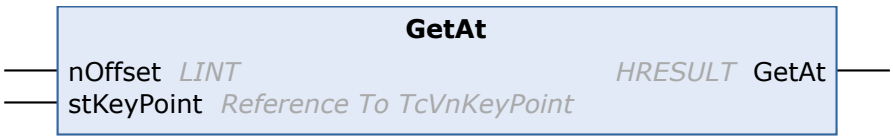
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 273]	ITcVnAccess_TcVnKeyPoint	Gets the value.
Set [▶ 274]	ITcVnAccess_TcVnKeyPoint	Sets the value.
GetAt [▶ 326]	ITcVnRandomAccess_TcVnKeyPoint	Gets the value
SetAt [▶ 327]	ITcVnRandomAccess_TcVnKeyPoint	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.8.1 GetAt



Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset    : LINT;
    stKeyPoint : Reference To TcVnKeyPoint;
END_VAR
```

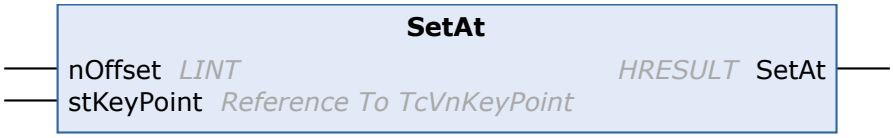
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
stKeyPoint	Reference To TcVnKeyPoint [▶ 244]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.8.2 SetAt



Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset    : LINT;
    stKeyPoint : Reference To TcVnKeyPoint;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
stKeyPoint	Reference To TcVnKeyPoint [▶ 244]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.9 ITcVnRandomAccess_TcVnPoint2_DINT

Offers a random access interface for TcVnPoint2_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnPoint2_DINT [▶ 274]
    ITcVnRandomAccess_TcVnPoint2_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 275]	ITcVnAccess_TcVnPoint2_DINT	Gets the value.
Set [▶ 275]	ITcVnAccess_TcVnPoint2_DINT	Sets the value.
GetAt [▶ 328]	ITcVnRandomAccess_TcVnPoint2_DINT	Gets the value
SetAt [▶ 328]	ITcVnRandomAccess_TcVnPoint2_DINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.9.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aPoint  : Reference To TcVnPoint2_DINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aPoint	Reference To TcVnPoint2_DINT [▶ 171]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.9.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aPoint  : Reference To TcVnPoint2_DINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aPoint	Reference To TcVnPoint2_DINT [▶ 171]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.10 ITcVnRandomAccess_TcVnPoint2_LREAL

Offers a random access interface for TcVnPoint2_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnPoint2_LREAL [▶ 276]
    ITcVnRandomAccess_TcVnPoint2_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 276]	ITcVnAccess_TcVnPoint2_LREAL	Gets the value
Set [▶ 277]	ITcVnAccess_TcVnPoint2_LREAL	Sets the value
GetAt [▶ 330]	ITcVnRandomAccess_TcVnPoint2_LREAL	Gets the value
SetAt [▶ 330]	ITcVnRandomAccess_TcVnPoint2_LREAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.10.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aPoint  : Reference To TcVnPoint2_LREAL;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aPoint	Reference To TcVnPoint2_LREAL [▶ 171]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.10.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aPoint  : Reference To TcVnPoint2_LREAL;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aPoint	Reference To TcVnPoint2_LREAL [▶ 171]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.11 ITcVnRandomAccess_TcVnPoint2_REAL

Offers a random access interface for TcVnPoint2_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnPoint2_REAL [▶ 277]
    ITcVnRandomAccess_TcVnPoint2_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 278]	ITcVnAccess_TcVnPoint2_REAL	Gets the value
Set [▶ 278]	ITcVnAccess_TcVnPoint2_REAL	Sets the value
GetAt [▶ 331]	ITcVnRandomAccess_TcVnPoint2_REAL	Gets the value
SetAt [▶ 332]	ITcVnRandomAccess_TcVnPoint2_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.11.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aPoint  : Reference To TcVnPoint2_REAL;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aPoint	Reference To TcVnPoint2_REAL [▶ 171]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.11.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aPoint  : Reference To TcVnPoint2_REAL;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aPoint	Reference To TcVnPoint2_REAL [▶ 171]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.12 ITcVnRandomAccess_TcVnPoint3_LREAL

Offers a random access interface for TcVnPoint3_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnPoint3_LREAL [▶ 279]
    ITcVnRandomAccess_TcVnPoint3_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 279]	ITcVnAccess_TcVnPoint3_LREAL	Gets the value
Set [▶ 279]	ITcVnAccess_TcVnPoint3_LREAL	Sets the value
GetAt [▶ 333]	ITcVnRandomAccess_TcVnPoint3_LREAL	Gets the value
SetAt [▶ 333]	ITcVnRandomAccess_TcVnPoint3_LREAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.12.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aPoint  : Reference To TcVnPoint3_LREAL;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aPoint	Reference To TcVnPoint3_LREAL [▶ 171]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.12.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aPoint  : Reference To TcVnPoint3_LREAL;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aPoint	Reference To TcVnPoint3_LREAL [▶ 171]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.13 ITcVnRandomAccess_TcVnPoint3_REAL

Offers a random access interface for TcVnPoint3_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnPoint3_REAL [▶ 280]
    ITcVnRandomAccess_TcVnPoint3_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 280]	ITcVnAccess_TcVnPoint3_REAL	Gets the value
Set [▶ 281]	ITcVnAccess_TcVnPoint3_REAL	Sets the value
GetAt [▶ 335]	ITcVnRandomAccess_TcVnPoint3_REAL	Gets the value
SetAt [▶ 335]	ITcVnRandomAccess_TcVnPoint3_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.13.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aPoint  : Reference To TcVnPoint3_REAL;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aPoint	Reference To TcVnPoint3_REAL [▶ 171]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.13.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aPoint  : Reference To TcVnPoint3_REAL;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aPoint	Reference To TcVnPoint3_REAL [▶ 171]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.14 ITcVnRandomAccess_TcVnRectangle_DINT

Offers a random access interface for TcVnRectangle_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnRectangle_DINT [▶ 281]
    ITcVnRandomAccess_TcVnRectangle_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 282]	ITcVnAccess_TcVnRectangle_DINT	Gets the value
Set [▶ 282]	ITcVnAccess_TcVnRectangle_DINT	Sets the value
GetAt [▶ 336]	ITcVnRandomAccess_TcVnRectangle_DINT	Gets the value
SetAt [▶ 337]	ITcVnRandomAccess_TcVnRectangle_DINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.14.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset     : LINT;
    stRectangle : Reference To TcVnRectangle_DINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
stRectangle	Reference To TcVnRectangle_DINT [▶ 257]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.14.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset     : LINT;
    stRectangle : Reference To TcVnRectangle_DINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
stRectangle	Reference To TcVnRectangle_DINT [▶ 257]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.15 ITcVnRandomAccess_TcVnVector2_DINT

Offers a random access interface for TcVnVector2_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector2_DINT [▶ 283]
    ITcVnRandomAccess_TcVnVector2_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 283]	ITcVnAccess_TcVnVector2_DINT	Gets the value
Set [▶ 284]	ITcVnAccess_TcVnVector2_DINT	Sets the value.
GetAt [▶ 338]	ITcVnRandomAccess_TcVnVector2_DINT	Gets the value
SetAt [▶ 338]	ITcVnRandomAccess_TcVnVector2_DINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.15.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector2_DINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector2_DINT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.15.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector2_DINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector2_DINT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.16 ITcVnRandomAccess_TcVnVector2_INT

Offers a random access interface for TcVnVector2_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector2_INT [▶ 284]
    ITcVnRandomAccess_TcVnVector2_INT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 285]	ITcVnAccess_TcVnVector2_INT	Gets the value
Set [▶ 285]	ITcVnAccess_TcVnVector2_INT	Sets the value.
GetAt [▶ 340]	ITcVnRandomAccess_TcVnVector2_INT	Gets the value
SetAt [▶ 340]	ITcVnRandomAccess_TcVnVector2_INT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.16.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector2_INT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector2_INT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.16.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector2_INT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector2_INT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.17 ITcVnRandomAccess_TcVnVector2_REAL

Offers a random access interface for TcVnVector2_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector2_REAL [▶ 286]
    ITcVnRandomAccess_TcVnVector2_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 286]	ITcVnAccess_TcVnVector2_REAL	Gets the value
Set [▶ 286]	ITcVnAccess_TcVnVector2_REAL	Sets the value
GetAt [▶ 341]	ITcVnRandomAccess_TcVnVector2_REAL	Gets the value
SetAt [▶ 342]	ITcVnRandomAccess_TcVnVector2_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.17.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector2_REAL;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector2_REAL [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.17.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector2_REAL;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector2_REAL [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.18 ITcVnRandomAccess_TcVnVector2_SINT

Offers a random access interface for TcVnVector2_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector2_SINT [▶ 287]
    ITcVnRandomAccess_TcVnVector2_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 287]	ITcVnAccess_TcVnVector2_SINT	Gets the value
Set [▶ 288]	ITcVnAccess_TcVnVector2_SINT	Sets the value.
GetAt [▶ 343]	ITcVnRandomAccess_TcVnVector2_SINT	Gets the value
SetAt [▶ 343]	ITcVnRandomAccess_TcVnVector2_SINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.18.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector2_SINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector2_SINT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.18.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector2_SINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector2_SINT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.19 ITcVnRandomAccess_TcVnVector2_UINT

Offers a random access interface for TcVnVector2_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector2_UINT [▶ 288]
    ITcVnRandomAccess_TcVnVector2_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 289]	ITcVnAccess_TcVnVector2_UINT	Gets the value
Set [▶ 289]	ITcVnAccess_TcVnVector2_UINT	Sets the value.
GetAt [▶ 345]	ITcVnRandomAccess_TcVnVector2_UINT	Gets the value
SetAt [▶ 345]	ITcVnRandomAccess_TcVnVector2_UINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.19.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector2_UINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector2_UINT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.19.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector2_UINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector2_UINT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.20 ITcVnRandomAccess_TcVnVector2_USINT

Offers a random access interface for TcVnVector2_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector2_USINT [▶ 290]
    ITcVnRandomAccess_TcVnVector2_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 290]	ITcVnAccess_TcVnVector2_USINT	Gets the value
Set [▶ 291]	ITcVnAccess_TcVnVector2_USINT	Sets the value.
GetAt [▶ 346]	ITcVnRandomAccess_TcVnVector2_USINT	Gets the value
SetAt [▶ 347]	ITcVnRandomAccess_TcVnVector2_USINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.20.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector2_USINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector2_USINT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.20.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector2_USINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector2_USINT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.21 ITcVnRandomAccess_TcVnVector3_INT

Offers a random access interface for TcVnVector3_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector3_INT [▶ 291]
    ITcVnRandomAccess_TcVnVector3_INT

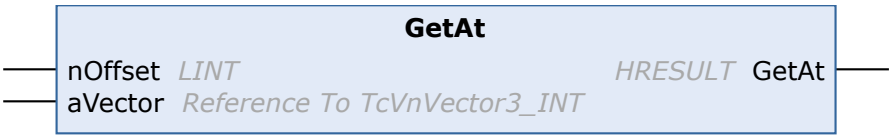
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 292]	ITcVnAccess_TcVnVector3_INT	Gets the value
Set [▶ 292]	ITcVnAccess_TcVnVector3_INT	Sets the value
GetAt [▶ 348]	ITcVnRandomAccess_TcVnVector3_INT	Gets the value
SetAt [▶ 348]	ITcVnRandomAccess_TcVnVector3_INT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.21.1 GetAt



Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector3_INT;
END_VAR
```

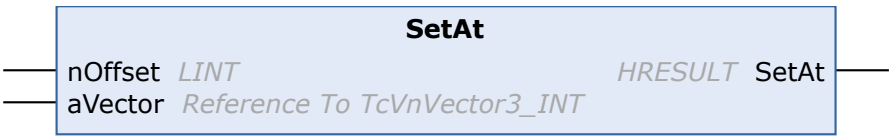
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector3_INT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.21.2 SetAt



Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector3_INT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector3_INT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.22 ITcVnRandomAccess_TcVnVector3_REAL

Offers a random access interface for TcVnVector3_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector3_REAL [▶ 293]
    ITcVnRandomAccess_TcVnVector3_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 293]	ITcVnAccess_TcVnVector3_REAL	Gets the value
Set [▶ 294]	ITcVnAccess_TcVnVector3_REAL	Sets the value
GetAt [▶ 350]	ITcVnRandomAccess_TcVnVector3_REAL	Gets the value
SetAt [▶ 350]	ITcVnRandomAccess_TcVnVector3_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.22.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector3_REAL;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector3_REAL [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.22.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector3_REAL;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector3_REAL [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.23 ITcVnRandomAccess_TcVnVector3_SINT

Offers a random access interface for TcVnVector3_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector3_SINT [▶ 294]
    ITcVnRandomAccess_TcVnVector3_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 295]	ITcVnAccess_TcVnVector3_SINT	Gets the value
Set [▶ 295]	ITcVnAccess_TcVnVector3_SINT	Sets the value
GetAt [▶ 351]	ITcVnRandomAccess_TcVnVector3_SINT	Gets the value
SetAt [▶ 352]	ITcVnRandomAccess_TcVnVector3_SINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.23.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector3_SINT;
END_VAR
```

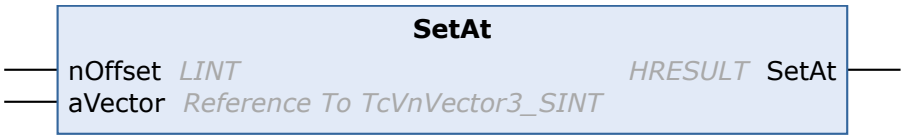
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector3_SINT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.23.2 SetAt



Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector3_SINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector3_SINT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.24 ITcVnRandomAccess_TcVnVector3_UINT

Offers a random access interface for TcVnVector3_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector3_UINT [▶ 296]
    ITcVnRandomAccess_TcVnVector3_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 296]	ITcVnAccess_TcVnVector3_UINT	Gets the value
Set [▶ 296]	ITcVnAccess_TcVnVector3_UINT	Sets the value
GetAt [▶ 353]	ITcVnRandomAccess_TcVnVector3_UINT	Gets the value
SetAt [▶ 353]	ITcVnRandomAccess_TcVnVector3_UINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.24.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector3_UINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector3_UINT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.24.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector3_UINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector3_UINT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.25 ITcVnRandomAccess_TcVnVector3_USINT

Offers a random access interface for TcVnVector3_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector3_USINT [▶ 297]
    ITcVnRandomAccess_TcVnVector3_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 297]	ITcVnAccess_TcVnVector3_USINT	Gets the value
Set [▶ 298]	ITcVnAccess_TcVnVector3_USINT	Sets the value
GetAt [▶ 355]	ITcVnRandomAccess_TcVnVector3_USINT	Gets the value
SetAt [▶ 355]	ITcVnRandomAccess_TcVnVector3_USINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.25.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector3_USINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector3_USINT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.25.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector3_USINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector3_USINT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.26 ITcVnRandomAccess_TcVnVector4_DINT

Offers a random access interface for TcVnVector4_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector4_DINT [▶ 298]
    ITcVnRandomAccess_TcVnVector4_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 299]	ITcVnAccess_TcVnVector4_DINT	Gets the value
Set [▶ 299]	ITcVnAccess_TcVnVector4_DINT	Sets the value
GetAt [▶ 356]	ITcVnRandomAccess_TcVnVector4_DINT	Gets the value
SetAt [▶ 357]	ITcVnRandomAccess_TcVnVector4_DINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.26.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_DINT;
END_VAR
```

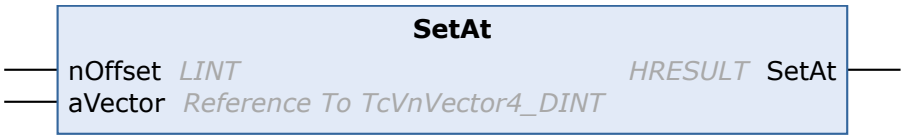
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector4_DINT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.26.2 SetAt



Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_DINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector4_DINT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.27 ITcVnRandomAccess_TcVnVector4_INT

Offers a random access interface for TcVnVector4_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector4_INT [▶ 300]
    ITcVnRandomAccess_TcVnVector4_INT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 300]	ITcVnAccess_TcVnVector4_INT	Gets the value
Set [▶ 301]	ITcVnAccess_TcVnVector4_INT	Sets the value
GetAt [▶ 358]	ITcVnRandomAccess_TcVnVector4_INT	Gets the value
SetAt [▶ 358]	ITcVnRandomAccess_TcVnVector4_INT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.27.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_INT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector4_INT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.27.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_INT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector4_INT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.28 ITcVnRandomAccess_TcVnVector4_LREAL

Offers a random access interface for TcVnVector4_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector4_LREAL [▶ 301]
    ITcVnRandomAccess_TcVnVector4_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 302]	ITcVnAccess_TcVnVector4_LREAL	Gets the value
Set [▶ 302]	ITcVnAccess_TcVnVector4_LREAL	Sets the value
GetAt [▶ 360]	ITcVnRandomAccess_TcVnVector4_LREAL	Gets the value
SetAt [▶ 360]	ITcVnRandomAccess_TcVnVector4_LREAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.28.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_LREAL;
END_VAR
```

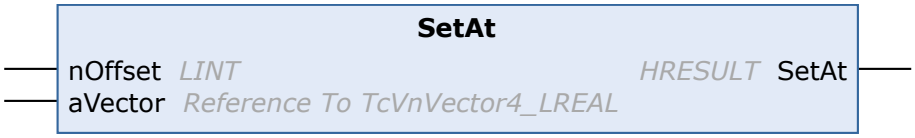
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector4_LREAL [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.28.2 SetAt



Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_LREAL;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector4_LREAL [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.29 ITcVnRandomAccess_TcVnVector4_SINT

Offers a random access interface for TcVnVector4_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector4_SINT [▶ 303]
    ITcVnRandomAccess_TcVnVector4_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 303]	ITcVnAccess_TcVnVector4_SINT	Gets the value
Set [▶ 304]	ITcVnAccess_TcVnVector4_SINT	Sets the value
GetAt [▶ 361]	ITcVnRandomAccess_TcVnVector4_SINT	Gets the value
SetAt [▶ 362]	ITcVnRandomAccess_TcVnVector4_SINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.29.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_SINT;
END_VAR
```

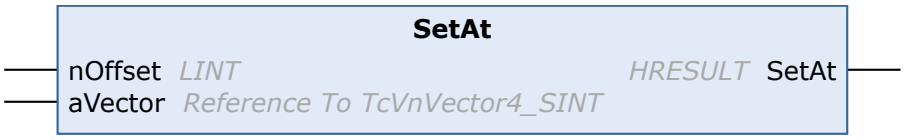
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector4_SINT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.29.2 SetAt



Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_SINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector4_SINT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.30 ITcVnRandomAccess_TcVnVector4_UINT

Offers a random access interface for TcVnVector4_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector4_UINT [▶ 304]
    ITcVnRandomAccess_TcVnVector4_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 304]	ITcVnAccess_TcVnVector4_UINT	Gets the value
Set [▶ 305]	ITcVnAccess_TcVnVector4_UINT	Sets the value
GetAt [▶ 363]	ITcVnRandomAccess_TcVnVector4_UINT	Gets the value
SetAt [▶ 363]	ITcVnRandomAccess_TcVnVector4_UINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.30.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_UINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector4_UINT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.30.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_UINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector4_UINT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.31 ITcVnRandomAccess_TcVnVector4_USINT

Offers a random access interface for TcVnVector4_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector4_USINT [▶ 305]
    ITcVnRandomAccess_TcVnVector4_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 306]	ITcVnAccess_TcVnVector4_USINT	Gets the value
Set [▶ 306]	ITcVnAccess_TcVnVector4_USINT	Sets the value
GetAt [▶ 365]	ITcVnRandomAccess_TcVnVector4_USINT	Gets the value
SetAt [▶ 365]	ITcVnRandomAccess_TcVnVector4_USINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.31.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_USINT;
END_VAR
```

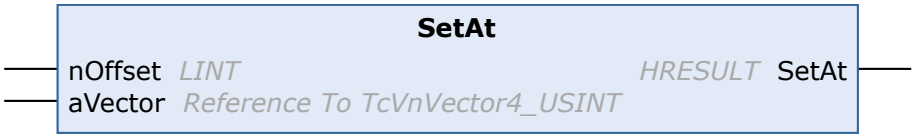
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector4_USINT [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.31.2 SetAt



Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_USINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector4_USINT [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.32 ITcVnRandomAccess_UDINT

Offers a random access interface for UDINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_UDINT [▶ 307]
    ITcVnRandomAccess_UDINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 307]	ITcVnAccess_UDINT	Gets the value
Set [▶ 308]	ITcVnAccess_UDINT	Sets the value
GetAt [▶ 366]	ITcVnRandomAccess_UDINT	Gets the value
SetAt [▶ 367]	ITcVnRandomAccess_UDINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.32.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue  : Reference To UDINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
nValue	Reference To UDINT	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.32.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue  : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
nValue	UDINT	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.33 ITcVnRandomAccess_UINT

Offers a random access interface for UINT values

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_UINT [▶ 308]
    ITcVnRandomAccess_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 309]	ITcVnAccess_UINT	Gets the value
Set [▶ 309]	ITcVnAccess_UINT	Sets the value
GetAt [▶ 368]	ITcVnRandomAccess_UINT	Gets the value
SetAt [▶ 368]	ITcVnRandomAccess_UINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.33.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue  : Reference To UINT;
END_VAR
```

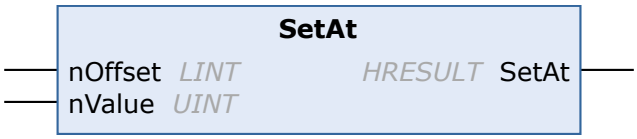
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
nValue	Reference To UINT	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.33.2 SetAt



Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue  : UINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
nValue	UINT	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.34 ITcVnRandomAccess_ULINT

Offers a randowm access interface for ULINT values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_ULINT [▶ 310]
    ITcVnRandomAccess_ULINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 310]	ITcVnAccess_ULINT	Gets the value
Set [▶ 310]	ITcVnAccess_ULINT	Sets the value
GetAt [▶ 369]	ITcVnRandomAccess_ULINT	Gets the value
SetAt [▶ 370]	ITcVnRandomAccess_ULINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.34.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue  : Reference To ULINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
nValue	Reference To ULINT	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.34.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue  : ULINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
nValue	ULINT	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.35 ITcVnRandomAccess_USINT

Offers a random access interface for USINT values

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_USINT [▶ 311]
    ITcVnRandomAccess_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 311]	ITcVnAccess_USINT	Gets the value
Set [▶ 312]	ITcVnAccess_USINT	Sets the value
GetAt [▶ 371]	ITcVnRandomAccess_USINT	Gets the value
SetAt [▶ 371]	ITcVnRandomAccess_USINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.35.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue  : Reference To USINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
nValue	Reference To USINT	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.35.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    nValue  : USINT;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
nValue	USINT	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.36 ITcVnRandomAccess_TcVnVector4_REAL

Offers a random access interface for TcVnVector4_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnVector4_REAL [▶ 312]
    ITcVnRandomAccess_TcVnVector4_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 313]	ITcVnAccess_TcVnVector4_REAL	Gets the value
Set [▶ 313]	ITcVnAccess_TcVnVector4_REAL	Sets the value
GetAt [▶ 372]	ITcVnRandomAccess_TcVnVector4_REAL	Gets the value
SetAt [▶ 373]	ITcVnRandomAccess_TcVnVector4_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.36.1 GetAt

Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_REAL;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector4_REAL [▶ 173]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.36.2 SetAt

Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset : LINT;
    aVector : Reference To TcVnVector4_REAL;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
aVector	Reference To TcVnVector4_REAL [▶ 173]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.3.37 ITcVnRandomAccess_TcVnRotatedRectangle

Offers a random access interface for TcVnRotatedRectangle values.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnAccess_TcVnRotatedRectangle [▶ 314]
    ITcVnRandomAccess_TcVnRotatedRectangle

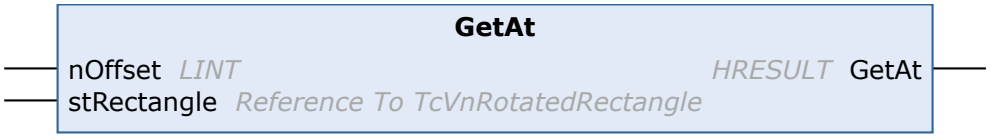
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Get [▶ 314]	ITcVnAccess_TcVnRotatedRectangle	Gets the value
Set [▶ 314]	ITcVnAccess_TcVnRotatedRectangle	Sets the value
GetAt [▶ 374]	ITcVnRandomAccess_TcVnRotatedRectangle	Gets the value
SetAt [▶ 374]	ITcVnRandomAccess_TcVnRotatedRectangle	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.3.37.1 GetAt



Gets the value

Syntax

Definition:

```
METHOD GetAt : HRESULT
VAR_INPUT
    nOffset     : LINT;
    stRectangle : Reference To TcVnRotatedRectangle;
END_VAR
```

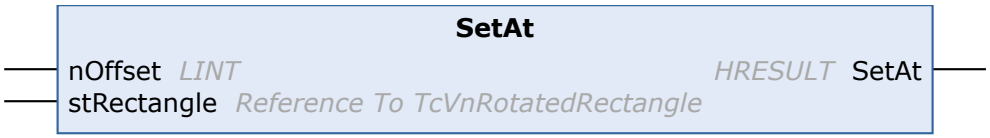
Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
stRectangle	Reference To TcVnRotatedRectangle [▶ 258]	Returns the value

Return value

HRESULT [▶ 150]

6.1.3.1.3.37.2 SetAt



Sets the value

Syntax

Definition:

```
METHOD SetAt : HRESULT
VAR_INPUT
    nOffset     : LINT;
    stRectangle : Reference To TcVnRotatedRectangle;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset to the current position
stRectangle	Reference To TcVnRotatedRectangle [▶ 258]	The value to set

Return value

HRESULT [▶ 150]

6.1.3.1.4 Iterators

Diese Gruppe enthält Interfaces zur Handhabung von Iteratoren, um auf Elemente von Containern [▶ 163] zuzugreifen.

6.1.3.1.4.1 ITcVnBidirectionalIterator

Offers an interface for a bidirectional iterator.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnIteratorBase [▶ 378]
    ITcVnForwardIterator [▶ 376]
      ITcVnBidirectionalIterator

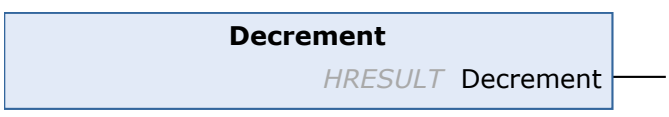
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
CheckIfBegin [▶ 379]	ITcVnIteratorBase	Checks if the iterator points to the first element.
CheckIfEnd [▶ 379]	ITcVnIteratorBase	Checks if the iterator points to the past-the-end element (alternatively use F_VN_CheckIfIteratorIsAtEnd).
GetValueSize [▶ 380]	ITcVnIteratorBase	Gets the memory size (in byte) required by the value the iterator points to.
GetValueTypeGuid [▶ 380]	ITcVnIteratorBase	Gets the type GUID of the value the iterator points to.
SetToBegin [▶ 380]	ITcVnIteratorBase	Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).
SetToEnd [▶ 381]	ITcVnIteratorBase	Sets the iterator to the past-the-end element.
CheckIfEqualTo [▶ 377]	ITcVnForwardIterator	Checks if iterator is equal to another iterator.
GetContainer [▶ 377]	ITcVnForwardIterator	Gets a pointer to the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). (Alternatively use F_VN_GetContainer.)
Increment [▶ 377]	ITcVnForwardIterator	Increments the iterator. (Alternatively use F_VN_IncrementIterator.)
SetContainer [▶ 378]	ITcVnForwardIterator	Sets the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use F_VN_SetContainer.)
Decrement [▶ 375]	ITcVnBidirectionalIterator	Decrements the iterator.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.4.1.1 Decrement



Decrements the iterator.

Syntax

Definition:

```
METHOD Decrement : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.1.4.2 ITcVnForwardIterator

Offers an interface for a forward iterator.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnIteratorBase [▶ 378]
    ITcVnForwardIterator

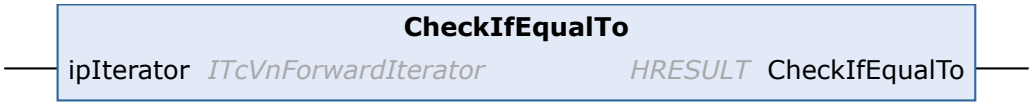
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
CheckIfBegin [▶ 379]	ITcVnIteratorBase	Checks if the iterator points to the first element.
CheckIfEnd [▶ 379]	ITcVnIteratorBase	Checks if the iterator points to the past-the-end element (alternatively use F_VN_CheckIfIteratorIsAtEnd).
GetValueSize [▶ 380]	ITcVnIteratorBase	Gets the memory size (in byte) required by the value the iterator points to.
GetValueTypeGuid [▶ 380]	ITcVnIteratorBase	Gets the type GUID of the value the iterator points to.
SetToBegin [▶ 380]	ITcVnIteratorBase	Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).
SetToEnd [▶ 381]	ITcVnIteratorBase	Sets the iterator to the past-the-end element.
CheckIfEqualTo [▶ 377]	ITcVnForwardIterator	Checks if iterator is equal to another iterator.
GetContainer [▶ 377]	ITcVnForwardIterator	Gets a pointer to the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). (Alternatively use F_VN_GetContainer.)
Increment [▶ 377]	ITcVnForwardIterator	Increments the iterator. (Alternatively use F_VN_IncrementIterator.)
SetContainer [▶ 378]	ITcVnForwardIterator	Sets the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use F_VN_SetContainer.)

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.4.2.1 CheckIfEqualTo



Checks if iterator is equal to another iterator.

Syntax

Definition:

```
METHOD CheckIfEqualTo : HRESULT
VAR_INPUT
    ipIterator : ITcVnForwardIterator;
END_VAR
```

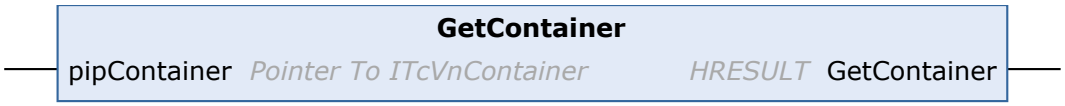
Inputs

Name	Type	Description
ipIterator	ITcVnForwardIterator [▶ 376]	Iterator interface to compare with

Return value

HRESULT [▶ 150]

6.1.3.1.4.2.2 GetContainer



Gets a pointer to the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). (Alternatively use F_VN_GetContainer.)

Syntax

Definition:

```
METHOD GetContainer : HRESULT
VAR_INPUT
    pipContainer : Pointer To ITcVnContainer;
END_VAR
```

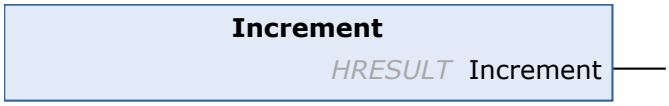
Inputs

Name	Type	Description
pipContainer	Pointer To ITcVnContainer [▶ 385]	Returns the container interface.

Return value

HRESULT [▶ 150]

6.1.3.1.4.2.3 Increment



Increments the iterator. (Alternatively use F_VN_IncrementIterator.)

Syntax

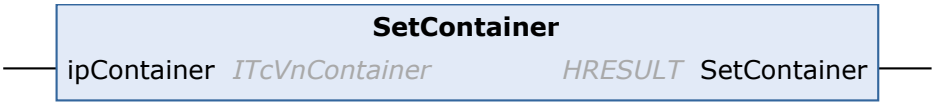
Definition:

```
METHOD Increment : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.1.4.2.4 SetContainer



Sets the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use F_VN_SetContainer.)

Syntax

Definition:

```
METHOD SetContainer : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container interface of which the content is to be assigned to the current element.

Return value

HRESULT [▶ 150]

6.1.3.1.4.3 ITcVnIteratorBase

Offers a base interface for iterators

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnIteratorBase

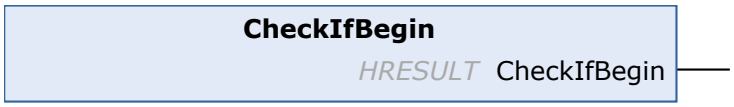
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
CheckIfBegin [▶ 379]	ITcVnIteratorBase	Checks if the iterator points to the first element.
CheckIfEnd [▶ 379]	ITcVnIteratorBase	Checks if the iterator points to the past-the-end element (alternatively use F_VN_CheckIfIteratorIsAtEnd).
GetValueSize [▶ 380]	ITcVnIteratorBase	Gets the memory size (in byte) required by the value the iterator points to.
GetValueTypeGuid [▶ 380]	ITcVnIteratorBase	Gets the type GUID of the value the iterator points to.
SetToBegin [▶ 380]	ITcVnIteratorBase	Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).
SetToEnd [▶ 381]	ITcVnIteratorBase	Sets the iterator to the past-the-end element.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.4.3.1 CheckIfBegin



Checks if the iterator points to the first element.

Syntax

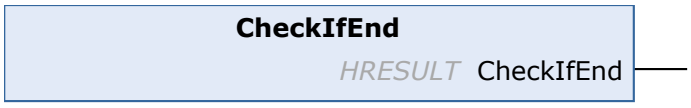
Definition:

```
METHOD CheckIfBegin : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.1.4.3.2 CheckIfEnd



Checks if the iterator points to the past-the-end element (alternatively use F_VN_CheckIfIteratorIsAtEnd).

Syntax

Definition:

```
METHOD CheckIfEnd : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.1.4.3.3 GetValueSize

Gets the memory size (in byte) required by the value the iterator points to.

Syntax

Definition:

```
METHOD GetValueSize : HRESULT
VAR_INPUT
    nSize : Reference To ULINT;
END_VAR
```

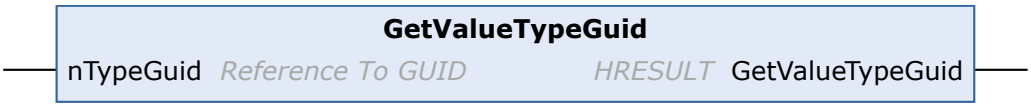
Inputs

Name	Type	Description
nSize	Reference To ULINT	Returns the element size in bytes.

Return value

HRESULT [▶ 150]

6.1.3.1.4.3.4 GetValueTypeGuid



Gets the type GUID of the value the iterator points to.

Syntax

Definition:

```
METHOD GetValueTypeGuid : HRESULT
VAR_INPUT
    nTypeGuid : Reference To GUID;
END_VAR
```

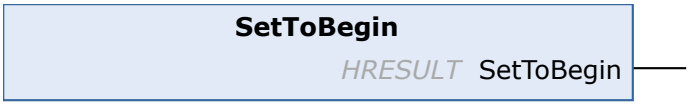
Inputs

Name	Type	Description
nTypeGuid	Reference To GUID	Returns the type GUID.

Return value

HRESULT [▶ 150]

6.1.3.1.4.3.5 SetToBegin



Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).

Syntax

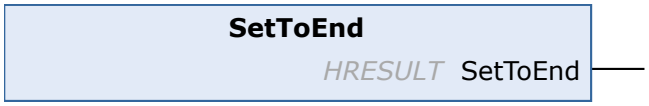
Definition:

```
METHOD SetToBegin : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.1.4.3.6 SetToEnd



Sets the iterator to the past-the-end element.

Syntax

Definition:

```
METHOD SetToEnd : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.1.4.4 ITcVnIteratorCopyCreator

Offers an interface providing a method for creating a new iterator to the same position.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnIteratorCopyCreator

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
Create [▶ 381]	ITcVnIteratorCopyCreator	Create a new iterator pointing to the position of the calling iterator.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.4.4.1 Create

Create a new iterator pointing to the position of the calling iterator.

Syntax

Definition:

```
METHOD Create : HRESULT
VAR_INPUT
    pipIterator : Pointer To ITcVnForwardIterator;
END_VAR
```

Inputs

Name	Type	Description
pipIterator	Pointer To ITcVnForwardIterator [▶ 376]	Returns the created iterator.

Return value

HRESULT [▶ 150]

6.1.3.1.4.5 ITcVnRandomAccessIterator

Offers an interface for a random access iterator.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnIteratorBase [▶ 378]
    ITcVnForwardIterator [▶ 376]
      ITcVnBidirectionalIterator [▶ 375]
        ITcVnRandomAccessIterator

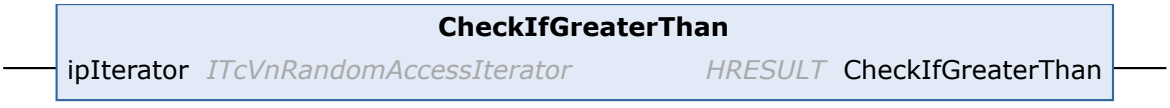
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
CheckIfBegin [▶ 379]	ITcVnIteratorBase	Checks if the iterator points to the first element.
CheckIfEnd [▶ 379]	ITcVnIteratorBase	Checks if the iterator points to the past-the-end element (alternatively use F_VN_CheckIfIteratorIsAtEnd).
GetValueSize [▶ 380]	ITcVnIteratorBase	Gets the memory size (in byte) required by the value the iterator points to.
GetValueTypeGuid [▶ 380]	ITcVnIteratorBase	Gets the type GUID of the value the iterator points to.
SetToBegin [▶ 380]	ITcVnIteratorBase	Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).
SetToEnd [▶ 381]	ITcVnIteratorBase	Sets the iterator to the past-the-end element.
CheckIfEqualTo [▶ 377]	ITcVnForwardIterator	Checks if iterator is equal to another iterator.
GetContainer [▶ 377]	ITcVnForwardIterator	Gets a pointer to the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). (Alternatively use F_VN_GetContainer.)
Increment [▶ 377]	ITcVnForwardIterator	Increments the iterator. (Alternatively use F_VN_IncrementIterator.)
SetContainer [▶ 378]	ITcVnForwardIterator	Sets the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use F_VN_SetContainer.)
Decrement [▶ 375]	ITcVnBidirectionalIterator	Decrements the iterator.
CheckIfGreaterThan [▶ 383]	ITcVnRandomAccessIterator	Checks if the iterator is greater than another iterator.
CheckIfLessThan [▶ 384]	ITcVnRandomAccessIterator	Checks if the iterator is less than another iterator.
GetContainerAt [▶ 384]	ITcVnRandomAccessIterator	Gets a pointer to the element at a specific offset from the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). (Alternatively use F_VN_GetAt_ITcVnContainer.)
SetContainerAt [▶ 385]	ITcVnRandomAccessIterator	Sets the element at a specific offset from the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use F_VN_SetAt_ITcVnContainer.)

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.4.5.1 CheckIfGreaterThan



Checks if the iterator is greater than another iterator.

Syntax

Definition:

```
METHOD CheckIfGreaterThan : HRESULT
VAR_INPUT
    ipIterator : ITcVnRandomAccessIterator;
END_VAR
```

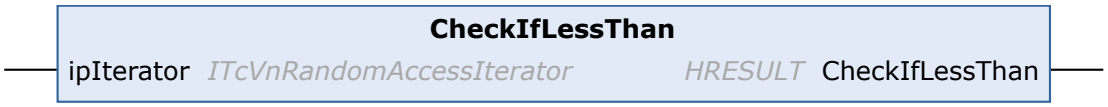
Inputs

Name	Type	Description
ipIterator	ITcVnRandomAccessIterator [▶ 382]	Iterator interface to compare with.

Return value

HRESULT [▶ 150]

6.1.3.1.4.5.2 CheckIfLessThan



Checks if the iterator is less than another iterator.

Syntax

Definition:

```
METHOD CheckIfLessThan : HRESULT
VAR_INPUT
    ipIterator : ITcVnRandomAccessIterator;
END_VAR
```

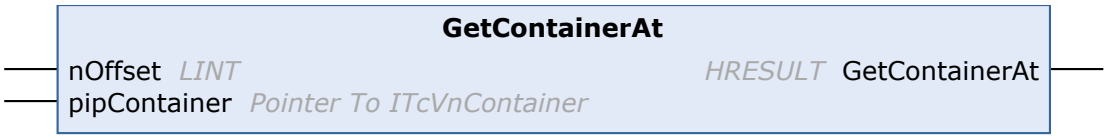
Inputs

Name	Type	Description
ipIterator	ITcVnRandomAccessIterator [▶ 382]	Iterator interface to compare with.

Return value

HRESULT [▶ 150]

6.1.3.1.4.5.3 GetContainerAt



Gets a pointer to the element at a specific offset from the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). (Alternatively use F_VN_GetAt_ITcVnContainer.)

Syntax

Definition:

```
METHOD GetContainerAt : HRESULT
VAR_INPUT
    nOffset      : LINT;
    pipContainer : Pointer To ITcVnContainer;
END_VAR
```

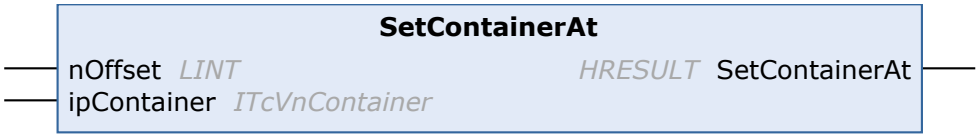
Inputs

Name	Type	Description
nOffset	LINT	Offset from the current element.
pipContainer	Pointer To ITcVnContainer [▶ 385]	Returns the pointer to the container interface.

Return value

HRESULT [▶ 150]

6.1.3.1.4.5.4 SetContainerAt



Sets the element at a specific offset from the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use F_VN_SetAt_ITcVnContainer.)

Syntax

Definition:

```
METHOD SetContainerAt : HRESULT
VAR_INPUT
    nOffset      : LINT;
    ipContainer : ITcVnContainer;
END_VAR
```

Inputs

Name	Type	Description
nOffset	LINT	Offset from the current element.
ipContainer	ITcVnContainer [▶ 385]	Container interface of which the content is to be assigned to the current element.

Return value

HRESULT [▶ 150]

6.1.3.1.5 ITcVnContainer

Offers an interface for an object container.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnContainer

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
CheckIfBasicContainer [▶ 387]	ITcVnContainer	Checks if the container contains only basic (non-container) elements.
CheckIfEmpty [▶ 387]	ITcVnContainer	Checks if the container is empty. (Alternatively use F_VN_CheckIfEmpty.)
GetBidirectionalIterator [▶ 387]	ITcVnContainer	Gets an interface pointer to a bidirectional iterator (if this iterator type is supported by the container) and increment its reference counter.
GetElementNum [▶ 388]	ITcVnContainer	Gets the size (number of elements) of the container. (Alternatively use F_VN_GetNumberOfElements.)
GetElementSize [▶ 388]	ITcVnContainer	Gets the size (in byte) of each element in the container.
GetElementTypeGuid [▶ 389]	ITcVnContainer	Gets the GUID of the container elements.
GetExportSize [▶ 389]	ITcVnContainer	Gets combined size (in byte) of all elements in the container.
GetForwardIterator [▶ 390]	ITcVnContainer	Gets an interface pointer to a forward iterator (if this iterator type is supported by the container) and increment its reference counter. (Alternatively use F_VN_GetForwardIterator.)
GetRandomAccessIterator [▶ 390]	ITcVnContainer	Gets an interface pointer to a random access iterator (if this iterator type is supported by the container) and increment its reference counter. (Alternatively use F_VN_GetRandomAccessIterator.)
GetTypeGuid [▶ 391]	ITcVnContainer	Gets GUID of the container.
GetTypeName [▶ 391]	ITcVnContainer	Gets the container type name as a string.

Weiterführende Informationen

Das Interface `ITcVnContainer` dient zur Handhabung von Containern [▶ 163]. Beachten Sie die Hinweise im Kapitel Interface Pointer [▶ 146].

Überschreiben von Containern

Grundsätzlich können Container mit API-Funktionen einfach überschrieben werden. Zurzeit ist ein Überschreiben allerdings nicht möglich, wenn der existierende Container und der zu schreibende Container abweichende Typen haben. Dies wird durch den Fehlercode 70E (INCOMPATIBLE) signalisiert.

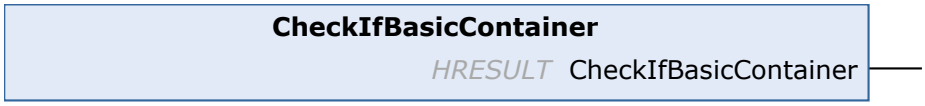
Verwandte Funktionen

- Basic Container Operations [▶ 564]
- Algebraic Container Operations [▶ 480]
- Container Statistics [▶ 891]
- Contour Analysis [▶ 985]

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.1.5.1 CheckIfBasicContainer



Checks if the container contains only basic (non-container) elements.

Syntax

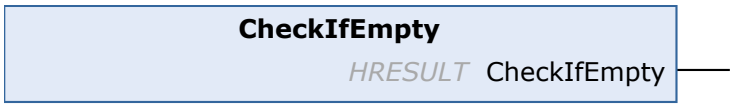
Definition:

```
METHOD CheckIfBasicContainer : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.1.5.2 CheckIfEmpty



Checks if the container is empty. (Alternatively use F_VN_CheckIfEmpty.)

Syntax

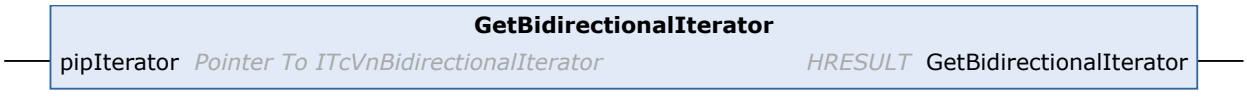
Definition:

```
METHOD CheckIfEmpty : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.1.5.3 GetBidirectionalIterator



Gets an interface pointer to a bidirectional iterator (if this iterator type is supported by the container) and increment its reference counter.

Syntax

Definition:

```
METHOD GetBidirectionalIterator : HRESULT
VAR_INPUT
    pipIterator : Pointer To ITcVnBidirectionalIterator;
END_VAR
```

Inputs

Name	Type	Description
pipIterator	Pointer To ITcVnBidirectionalIterator [▶ 375]	Returns the iterator interface.

Return value

HRESULT [▶ 150]

6.1.3.1.5.4 GetElementNum

Gets the size (number of elements) of the container. (Alternatively use F_VN_GetNumberOfElements.)

Syntax

Definition:

```
METHOD GetElementNum : HRESULT
VAR_INPUT
    nElementNum : Reference To ULINT;
END_VAR
```

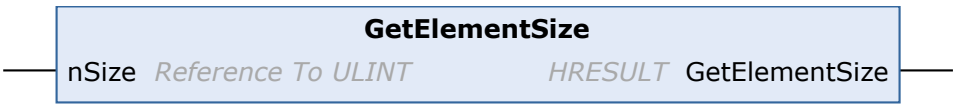
Inputs

Name	Type	Description
nElementNum	Reference To ULINT	Returns the number of elements in the container.

Return value

HRESULT [▶ 150]

6.1.3.1.5.5 GetElementSize



Gets the size (in byte) of each element in the container.

Syntax

Definition:

```
METHOD GetElementSize : HRESULT
VAR_INPUT
    nSize : Reference To ULINT;
END_VAR
```

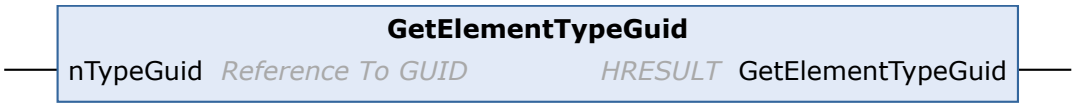
Inputs

Name	Type	Description
nSize	Reference To ULINT	Returns the size in byte of a single element in the container.

Return value

HRESULT [▶ 150]

6.1.3.1.5.6 GetElementTypeGuid



Gets the GUID of the container elements.

Syntax

Definition:

```
METHOD GetElementTypeGuid : HRESULT
VAR_INPUT
    nTypeGuid : Reference To GUID;
END_VAR
```

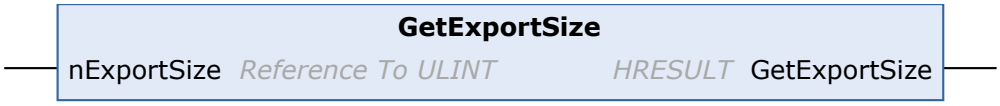
Inputs

Name	Type	Description
nTypeGuid	Reference To GUID	Returns the GUID of the container elements.

Return value

HRESULT [▶ 150]

6.1.3.1.5.7 GetExportSize



Gets combined size (in byte) of all elements in the container.

Syntax

Definition:

```
METHOD GetExportSize : HRESULT
VAR_INPUT
    nExportSize : Reference To ULINT;
END_VAR
```

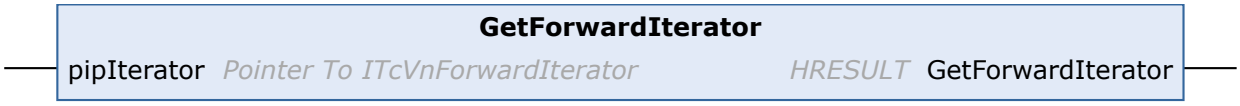
Inputs

Name	Type	Description
nExportSize	Reference To ULINT	Returns the combined size (in byte) of all elements in the container.

Return value

HRESULT [▶ 150]

6.1.3.1.5.8 GetForwardIterator



Gets an interface pointer to a forward iterator (if this iterator type is supported by the container) and increment its reference counter. (Alternatively use F_VN_GetForwardIterator.)

Syntax

Definition:

```
METHOD GetForwardIterator : HRESULT
VAR_INPUT
    pipIterator : Pointer To ITcVnForwardIterator;
END_VAR
```

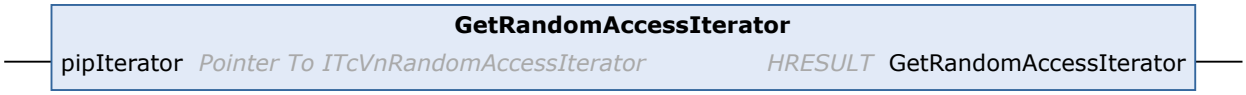
Inputs

Name	Type	Description
pipIterator	Pointer To ITcVnForwardIterator [▶ 376]	Returns the iterator interface.

Return value

HRESULT [▶ 150]

6.1.3.1.5.9 GetRandomAccessIterator



Gets an interface pointer to a random access iterator (if this iterator type is supported by the container) and increment its reference counter. (Alternatively use F_VN_GetRandomAccessIterator.)

Syntax

Definition:

```
METHOD GetRandomAccessIterator : HRESULT
VAR_INPUT
    pipIterator : Pointer To ITcVnRandomAccessIterator;
END_VAR
```

Inputs

Name	Type	Description
pipIterator	Pointer To ITcVnRandomAccessIterator [▶ 382]	Returns the iterator interface.

Return value

HRESULT [▶ 150]

6.1.3.1.5.10 GetTypeGuid

Gets GUID of the container.

Syntax

Definition:

```
METHOD GetTypeGuid : HRESULT
VAR_INPUT
    nTypeGuid : Reference To GUID;
END_VAR
```

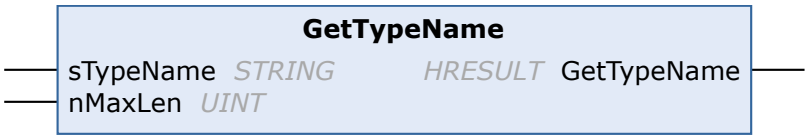
Inputs

Name	Type	Description
nTypeGuid	Reference To GUID	Returns the GUID of the container.

Return value

HRESULT [▶ 150]

6.1.3.1.5.11 GetTypeName



Gets the container type name as a string.

Syntax

Definition:

```
METHOD GetTypeName : HRESULT
VAR_INPUT
    sTypeName : STRING;
    nMaxLen   : UINT;
END_VAR
```

Inputs

Name	Type	Description
sTypeName	STRING	Returns the container type name as a string.
nMaxLen	UINT	Maximum string length allowed to be written in sTypeName

Return value

HRESULT [▶ 150]

6.1.3.2 Images

Diese Gruppe enthält Interfaces zur Aufnahme und Handhabung von Bildern [▶ 159].

6.1.3.2.1 Acquisition

Diese Gruppe enthält Interfaces zur Handhabung der Bildaufnahme.

6.1.3.2.1.1 ITcIoFileImageAcquisition

Interface for file image acquisition.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcIoImageAcquisition [▶ 396]
    ITcIoFileImageAcquisition

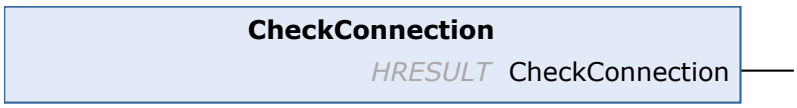
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
RegisterReceiver [▶ 396]	ITcIoImageAcquisition	Register callback interface.
UnregisterReceiver [▶ 397]	ITcIoImageAcquisition	Unregister callback interface.
OpenCamera [▶ 397]	ITcIoImageAcquisition	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 397]	ITcIoImageAcquisition	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 398]	ITcIoImageAcquisition	Start the image acquisition.
StopAcquisition [▶ 398]	ITcIoImageAcquisition	Stop the image acquisition.
SoftwareTrigger [▶ 398]	ITcIoImageAcquisition	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 399]	ITcIoImageAcquisition	Send initialization commands to the camera.
CheckConnection [▶ 392]	ITcIoFileImageAcquisition	Checks the connection
TriggerImage [▶ 393]	ITcIoFileImageAcquisition	Initialize the software trigger and trigger a single image. Skips nSkipImages and triggers the capturing of image nSkipImages+1.
TriggerImageByName [▶ 393]	ITcIoFileImageAcquisition	Initialize the software trigger and trigger a single image specified by its name in the client assistant.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.1.1.1 CheckConnection



Checks the connection

Syntax

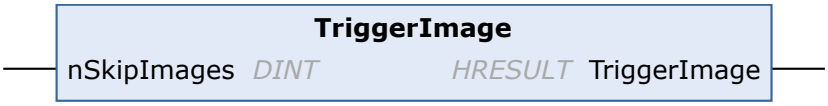
Definition:

```
METHOD CheckConnection : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.2.1.1.2 TriggerImage



Initialize the software trigger and trigger a single image. Skips nSkipImages and triggers the capturing of image nSkipImages+1.

Syntax

Definition:

```
METHOD TriggerImage : HRESULT
VAR_INPUT
    nSkipImages : DINT;
END_VAR
```

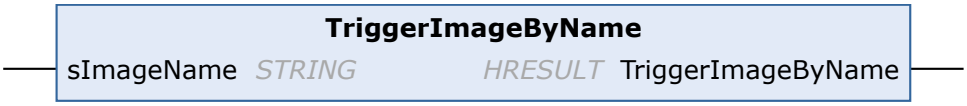
Inputs

Name	Type	Description
nSkipImages	DINT	Number of images to skip

Return value

HRESULT [▶ 150]

6.1.3.2.1.1.3 TriggerImageByName



Initialize the software trigger and trigger a single image specified by its name in the client assistant.

Syntax

Definition:

```
METHOD TriggerImageByName : HRESULT
VAR_INPUT
    sImageName : STRING;
END_VAR
```

Inputs

Name	Type	Description
sImageName	STRING	Image name to trigger

Return value

HRESULT [▶ 150]

6.1.3.2.1.2 ITcIoFileImageRecv

Interface for a image receiver.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcIoImageRecv [▶ 399]
    ITcIoFileImageRecv

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
ReceiveImage [▶ 400]	ITcIoImageRecv	Receive an image from an instance of ITcIoGevImageAcquisition.
ReceiveOpResult [▶ 400]	ITcIoImageRecv	Receive an operation result from an instance of ITcIoGevImageAcquisition.
ReceiveImage [▶ 394]	ITcIoFileImageRecv	Receive an image from an instance of ITcIoFileImageAcquisition.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.1.2.1 ReceiveImage

Receive an image from an instance of ITcIoFileImageAcquisition.

Syntax

Definition:

```
METHOD ReceiveImage : HRESULT
VAR_INPUT
    ipImage             : ITcVnImageBase;
    sFileName           : STRING;
    hrAcquisitionResult : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImageBase [▶ 429]	Interface pointer to the image.
sFileName	STRING	Returns the filename to the image.
hrAcquisitionResult	HRESULT [▶ 150]	HRESULT indicating the status of the image acquisition.

Return value

HRESULT [▶ 150]

6.1.3.2.1.3 ITcIoGevImageAcquisition

Interface for GigE Vision image acquisition.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcIoImageAcquisition [▶ 396]
    ITcIoGevImageAcquisition

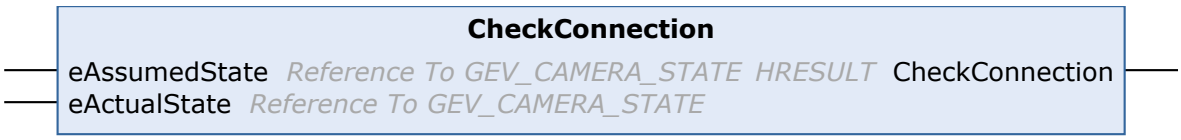
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
RegisterReceiver [▶ 396]	ITcIoImageAcquisition	Register callback interface.
UnregisterReceiver [▶ 397]	ITcIoImageAcquisition	Unregister callback interface.
OpenCamera [▶ 397]	ITcIoImageAcquisition	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 397]	ITcIoImageAcquisition	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 398]	ITcIoImageAcquisition	Start the image acquisition.
StopAcquisition [▶ 398]	ITcIoImageAcquisition	Stop the image acquisition.
SoftwareTrigger [▶ 398]	ITcIoImageAcquisition	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 399]	ITcIoImageAcquisition	Send initialization commands to the camera.
CheckConnection [▶ 395]	ITcIoGevImageAcquisition	Checks the camera connection

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.1.3.1 CheckConnection



Checks the camera connection

Syntax

Definition:

```
METHOD CheckConnection : HRESULT
VAR_INPUT
    eAssumedState : Reference To GEV_CAMERA_STATE;
    eActualState  : Reference To GEV_CAMERA_STATE;
END_VAR
```

Inputs

Name	Type	Description
eAssumedState	Reference To GEV_CAMERA_STATE [▶ 236]	The internally assumed state of the camera.
eActualState	Reference To GEV_CAMERA_STATE [▶ 236]	The actually observable state of the camera.

Return value

HRESULT [▶ 150]

6.1.3.2.1.4 ITcIoImageAcquisition

Interface for image acquisition.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcIoImageAcquisition

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
RegisterReceiver [▶ 396]	ITcIoImageAcquisition	Register callback interface.
UnregisterReceiver [▶ 397]	ITcIoImageAcquisition	Unregister callback interface.
OpenCamera [▶ 397]	ITcIoImageAcquisition	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 397]	ITcIoImageAcquisition	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 398]	ITcIoImageAcquisition	Start the image acquisition.
StopAcquisition [▶ 398]	ITcIoImageAcquisition	Stop the image acquisition.
SoftwareTrigger [▶ 398]	ITcIoImageAcquisition	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 399]	ITcIoImageAcquisition	Send initialization commands to the camera.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.1.4.1 RegisterReceiver

Register callback interface.

Syntax

Definition:

```
METHOD RegisterReceiver : HRESULT
VAR_INPUT
    ipRecv : ITcIoImageRecv;
END_VAR
```

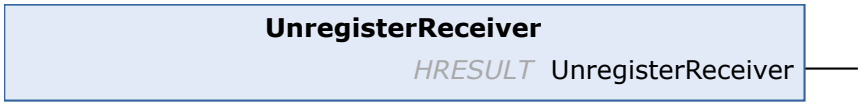
Inputs

Name	Type	Description
ipRecv	ITcIoImageRecv [▶ 399]	Pointer to an interface containing the callback function.

Return value

HRESULT [▶ 150]

6.1.3.2.1.4.2 UnregisterReceiver



Unregister callback interface.

Syntax

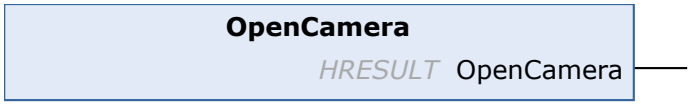
Definition:

```
METHOD UnregisterReceiver : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.2.1.4.3 OpenCamera



Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).

Syntax

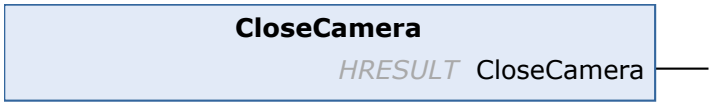
Definition:

```
METHOD OpenCamera : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.2.1.4.4 CloseCamera



Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).

Syntax

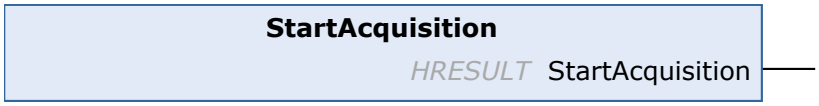
Definition:

```
METHOD CloseCamera : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.2.1.4.5 StartAcquisition



Start the image acquisition.

Syntax

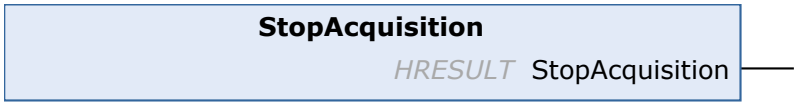
Definition:

```
METHOD StartAcquisition : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.2.1.4.6 StopAcquisition



Stop the image acquisition.

Syntax

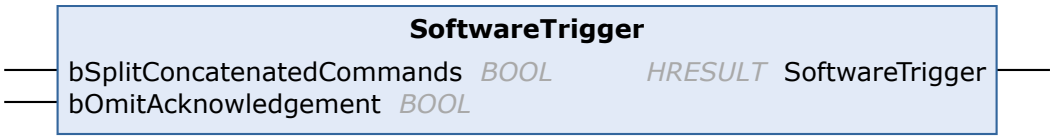
Definition:

```
METHOD StopAcquisition : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.2.1.4.7 SoftwareTrigger



Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.

Syntax

Definition:

```
METHOD SoftwareTrigger : HRESULT
VAR_INPUT
    bSplitConcatenatedCommands : BOOL;
    bOmitAcknowledgement       : BOOL;
END_VAR
```

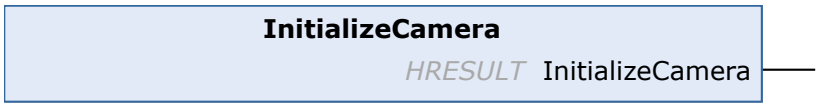
Inputs

Name	Type	Description
bSplitConcatenatedCommands	BOOL	If true, multi-read and multi-write commands are split into sequences of single read and single write commands, respectively.
bOmitAcknowledgement	BOOL	Indicates that no acknowledge packet should be requested.

Return value

HRESULT [▶ 150]

6.1.3.2.1.4.8 InitializeCamera



Send initialization commands to the camera.

Syntax

Definition:

```
METHOD InitializeCamera : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.2.1.5 ITcIoImageRecv

Interface for a image receiver.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcIoImageRecv

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
ReceiveImage [▶ 400]	ITcIoImageRecv	Receive an image from an instance of ITcIoGevImageAcquisition.
ReceiveOpResult [▶ 400]	ITcIoImageRecv	Receive an operation result from an instance of ITcIoGevImageAcquisition.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.1.5.1 ReceiveImage

Receive an image from an instance of ITcIoGevImageAcquisition.

Syntax

Definition:

```
METHOD ReceiveImage : HRESULT
VAR_INPUT
    ipImage             : ITcVnImageBase;
    hrAcquisitionResult : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImageBase [▶ 429]	Interface pointer to the image.
hrAcquisitionResult	HRESULT [▶ 150]	HRESULT indicating the status of the image acquisition.

Return value

HRESULT [▶ 150]

6.1.3.2.1.5.2 ReceiveOpResult

Receive an operation result from an instance of ITcIoGevImageAcquisition.

Syntax

Definition:

```
METHOD ReceiveOpResult : HRESULT
VAR_INPUT
    hrOperationResult : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrOperationResult	HRESULT [▶ 150]	HRESULT indicating the status of the performed operation.

Return value

HRESULT [▶ 150]

6.1.3.2.1.6 ITcVnFileImageProvider

Interface for an image provider for images from outside realtime environment.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnImageProvider [▶ 405]
    ITcVnFileImageProvider

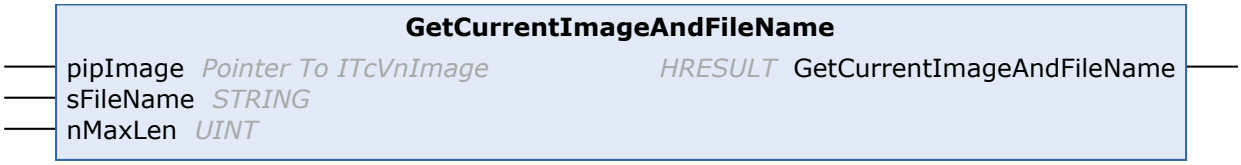
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
CheckIfBusy [▶ 406]	ITcVnImageProvider	Check whether the module is busy; i.e, an operation is in progress.
CheckIfCameraConnected [▶ 406]	ITcVnImageProvider	Check if the camera is properly connected (Returns S_OK if connected and S_FALSE if not. Can only check software connection, i.e. the check is unable to determine if the cable is connected or not in case that S_FALSE is returned).
GetCurrentImage [▶ 407]	ITcVnImageProvider	Gets the current image by detaching its internal reference.
GetError [▶ 407]	ITcVnImageProvider	Gets result of the last operation performed.
OpenCamera [▶ 408]	ITcVnImageProvider	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 408]	ITcVnImageProvider	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 408]	ITcVnImageProvider	Start image acquisition.
StopAcquisition [▶ 408]	ITcVnImageProvider	Stop image acquisition.
SoftwareTrigger [▶ 409]	ITcVnImageProvider	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 409]	ITcVnImageProvider	Set camera to initial state.
GetCurrentImageAndFileName [▶ 402]	ITcVnFileImageProvider	Get the current image by detaching its internal reference.
TriggerImage [▶ 402]	ITcVnFileImageProvider	Initialize the software trigger and trigger a single image. Skips nSkipImages and triggers the capturing of image nSkipImages+1.
TriggerImageByName [▶ 403]	ITcVnFileImageProvider	Initialize the software trigger and trigger a single image specified by its name in the client assistant.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.1.6.1 GetCurrentImageAndFileName



Get the current image by detaching its internal reference.

Syntax

Definition:

```
METHOD GetCurrentImageAndFileName : HRESULT
VAR_INPUT
    pipImage  : Pointer To ITcVnImage;
    sFileName : STRING;
    nMaxLen   : UINT;
END_VAR
```

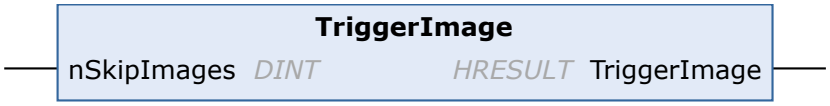
Inputs

Name	Type	Description
pipImage	Pointer To ITcVnImage [▶ 426]	pointer to the interface pointer to be returned, might be 0 if the image acquisition failed
sFileName	STRING	returns the filename as a string
nMaxLen	UINT	maximum string length allowed to be written in sFileName

Return value

HRESULT [▶ 150]

6.1.3.2.1.6.2 TriggerImage



Initialize the software trigger and trigger a single image. Skips nSkipImages and triggers the capturing of image nSkipImages+1.

Syntax

Definition:

```
METHOD TriggerImage : HRESULT
VAR_INPUT
    nSkipImages : DINT;
END_VAR
```

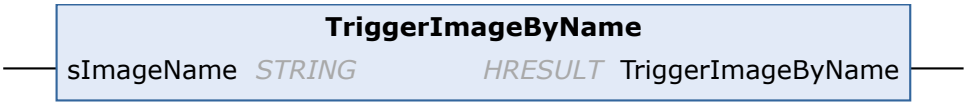
Inputs

Name	Type	Description
nSkipImages	DINT	Number of images to skip

Return value

HRESULT [▶ 150]

6.1.3.2.1.6.3 TriggerImageByName



Initialize the software trigger and trigger a single image specified by its name in the client assistant.

Syntax

Definition:

```
METHOD TriggerImageByName : HRESULT
VAR_INPUT
    sImageName : STRING;
END_VAR
```

Inputs

Name	Type	Description
sImageName	STRING	Image name to trigger

Return value

HRESULT [▶ 150]

6.1.3.2.1.7 ITcVnGevImageProvider

Interface for a GigE Vision image provider.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnImageProvider [▶ 405]
    ITcVnGevImageProvider

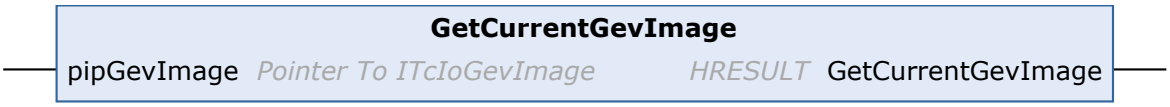
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
CheckIfBusy [▶ 406]	ITcVnImageProvider	Check whether the module is busy; i.e, an operation is in progress.
CheckIfCameraConnected [▶ 406]	ITcVnImageProvider	Check if the camera is properly connected (Returns S_OK if connected and S_FALSE if not. Can only check software connection, i.e. the check is unable to determine if the cable is connected or not in case that S_FALSE is returned).
GetCurrentImage [▶ 407]	ITcVnImageProvider	Gets the current image by detaching its internal reference.
GetError [▶ 407]	ITcVnImageProvider	Gets result of the last operation performed.
OpenCamera [▶ 408]	ITcVnImageProvider	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 408]	ITcVnImageProvider	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 408]	ITcVnImageProvider	Start image acquisition.
StopAcquisition [▶ 408]	ITcVnImageProvider	Stop image acquisition.
SoftwareTrigger [▶ 409]	ITcVnImageProvider	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 409]	ITcVnImageProvider	Set camera to initial state.
GetCurrentGevImage [▶ 404]	ITcVnGevImageProvider	Gets the current GigE Vision image by detaching its internal reference.
GetCurrentImageWithGvspInfo [▶ 405]	ITcVnGevImageProvider	Gets the current image by detaching its internal reference and additionally provide the GVSP info.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.1.7.1 GetCurrentGevImage



Gets the current GigE Vision image by detaching its internal reference.

Syntax

Definition:

```
METHOD GetCurrentGevImage : HRESULT
VAR_INPUT
    pipGevImage : Pointer To ITcIoGevImage;
END_VAR
```

Inputs

Name	Type	Description
pipGevImage	Pointer To ITcIoGevImage [▶ 422]	Pointer to the interface pointer to be returned, might be 0 if the image acquisition failed.

Return value

HRESULT [▶ 150]

6.1.3.2.1.7.2 GetCurrentImageWithGvspInfo

Gets the current image by detaching its internal reference and additionally provide the GVSP info.

Syntax

Definition:

```
METHOD GetCurrentImageWithGvspInfo : HRESULT
VAR_INPUT
    pipImage   : Pointer To ITcVnImage;
    stGvspInfo : Reference To GVSP_IMAGE_INFO;
END_VAR
```

Inputs

Name	Type	Description
pipImage	Pointer To ITcVnImage [▶ 426]	Pointer to the interface pointer to be returned, might be 0 if the image acquisition failed.
stGvspInfo	Reference To GVSP_IMAGE_INFO [▶ 238]	Contains useful meta information like image id, time stamp etc.

Return value

HRESULT [▶ 150]

6.1.3.2.1.8 ITcVnImageProvider

Interface for an image provider.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnImageProvider

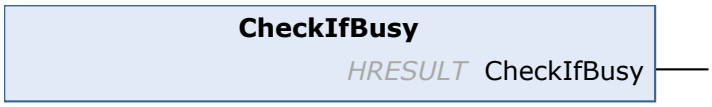
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
CheckIfBusy [▶ 406]	ITcVnImageProvider	Check whether the module is busy; i.e, an operation is in progress.
CheckIfCameraConnected [▶ 406]	ITcVnImageProvider	Check if the camera is properly connected (Returns S_OK if connected and S_FALSE if not. Can only check software connection, i.e. the check is unable to determine if the cable is connected or not in case that S_FALSE is returned).
GetCurrentImage [▶ 407]	ITcVnImageProvider	Gets the current image by detaching its internal reference.
GetError [▶ 407]	ITcVnImageProvider	Gets result of the last operation performed.
OpenCamera [▶ 408]	ITcVnImageProvider	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 408]	ITcVnImageProvider	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 408]	ITcVnImageProvider	Start image acquisition.
StopAcquisition [▶ 408]	ITcVnImageProvider	Stop image acquisition.
SoftwareTrigger [▶ 409]	ITcVnImageProvider	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 409]	ITcVnImageProvider	Set camera to initial state.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.1.8.1 CheckIfBusy



Check whether the module is busy; i.e, an operation is in progress.

Syntax

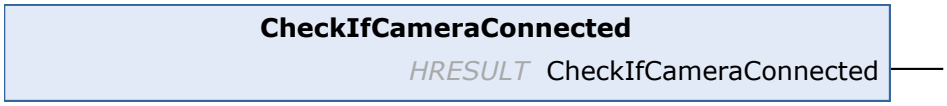
Definition:

```
METHOD CheckIfBusy : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.2.1.8.2 CheckIfCameraConnected



Check if the camera is properly connected (Returns S_OK if connected and S_FALSE if not. Can only check software connection, i.e. the check is unable to determine if the cable is connected or not in case that S_FALSE is returned).

Syntax

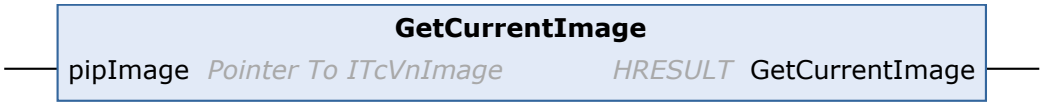
Definition:

```
METHOD CheckIfCameraConnected : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.2.1.8.3 GetCurrentImage



Gets the current image by detaching its internal reference.

Syntax

Definition:

```
METHOD GetCurrentImage : HRESULT
VAR_INPUT
    pipImage : Pointer To ITcVnImage;
END_VAR
```

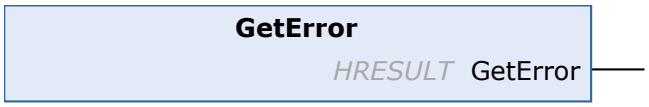
Inputs

Name	Type	Description
pipImage	Pointer To ITcVnImage [▶ 426]	Pointer to the interface pointer to be returned, might be 0 if the image acquisition failed.

Return value

HRESULT [▶ 150]

6.1.3.2.1.8.4 GetError



Gets result of the last operation performed.

Syntax

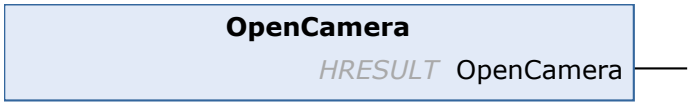
Definition:

```
METHOD GetError : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.2.1.8.5 OpenCamera



Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).

Syntax

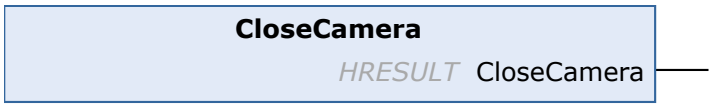
Definition:

```
METHOD OpenCamera : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.2.1.8.6 CloseCamera



Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).

Syntax

Definition:

```
METHOD CloseCamera : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.2.1.8.7 StartAcquisition

Start image acquisition.

Syntax

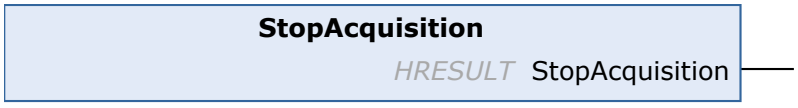
Definition:

```
METHOD StartAcquisition : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.2.1.8.8 StopAcquisition



Stop image acquisition.

Syntax

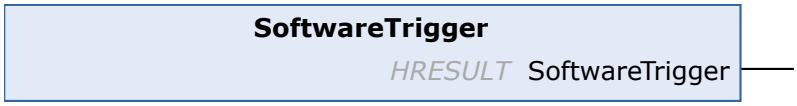
Definition:

```
METHOD StopAcquisition : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.2.1.8.9 SoftwareTrigger



Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.

Syntax

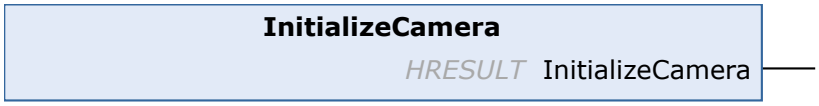
Definition:

```
METHOD SoftwareTrigger : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.2.1.8.10 InitializeCamera



Set camera to initial state.

Syntax

Definition:

```
METHOD InitializeCamera : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.2.2 Export

Diese Gruppe enthält Interfaces zur Handhabung des Daten-Exports im Zusammenhang mit Bildern.

6.1.3.2.2.1 ITcVnBitmapExport

Interface for exporting an image as a Windows Bitmap (BMP).

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnBitmapExport

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
GetBitmapSize [▶ 410]	ITcVnBitmapExport	Gets size of the image if it is converted to a Windows Bitmap (BMP).
GetBitmapImage [▶ 411]	ITcVnBitmapExport	Export the image as a Windows Bitmap (BMP) into a given buffer.

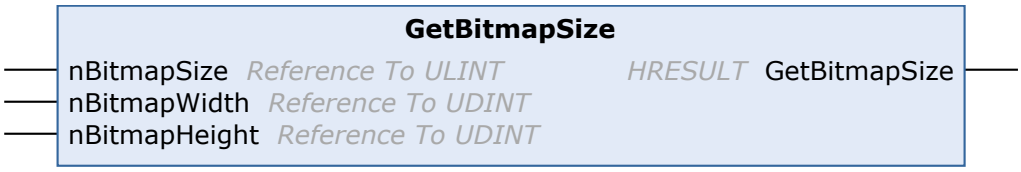
Weiterführende Informationen

Ein Bitmap kann per Definition nur 8 Bit (für 1-Kanal Bilder) oder 24 Bit (für 3- und 4-Kanal Bilder) haben. Daher werden 16-Bit Bilder sowie Bilder mit Element-Typ REAL oder LREAL für den Bitmap-Export konvertiert. Bei 16-Bit Bildern wird der Wertebereich einfach auf 8 Bit herunterskaliert (0 entspricht 0 und 255 entspricht 65.280). Bei (L)REAL-Bildern wird der Wertebereich [-1, +1] auf [-127, +127] skaliert und als signed Integer übertragen. Werte außerhalb dieser Grenzen werden dabei als -1 bzw. +1 interpretiert.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.2.1.1 GetBitmapSize



Gets size of the image if it is converted to a Windows Bitmap (BMP).

Syntax

Definition:

```
METHOD GetBitmapSize : HRESULT
VAR_INPUT
    nBitmapSize   : Reference To ULINT;
    nBitmapWidth  : Reference To UDINT;
    nBitmapHeight : Reference To UDINT;
END_VAR
```

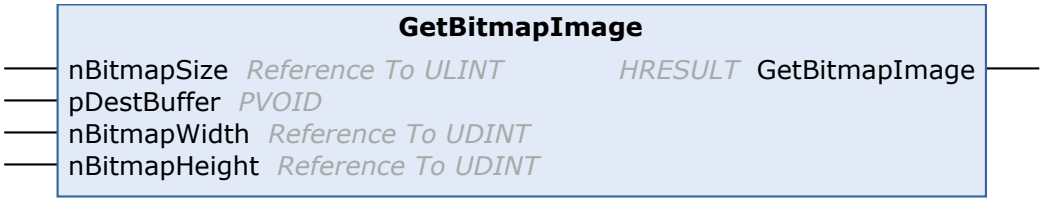
Inputs

Name	Type	Description
nBitmapSize	Reference To ULINT	Output parameter containing the required buffer size.
nBitmapWidth	Reference To UDINT	Desired width or 0 to keep the original width (in) and actual width (out).
nBitmapHeight	Reference To UDINT	Desired height or 0 to keep the original height (in) and actual height (out).

Return value

HRESULT [▶ 150]

6.1.3.2.2.1.2 GetBitmapImage



Export the image as a Windows Bitmap (BMP) into a given buffer.

Syntax

Definition:

```
METHOD GetBitmapImage : HRESULT
VAR_INPUT
    nBitmapSize   : Reference To ULINT;
    pDestBuffer   : PVOID;
    nBitmapWidth  : Reference To UDINT;
    nBitmapHeight : Reference To UDINT;
END_VAR
```

Inputs

Name	Type	Description
nBitmapSize	Reference To ULINT	Maximum buffer size (in) and actual buffer size (out).
pDestBuffer	PVOID	Pointer to the destination buffer.
nBitmapWidth	Reference To UDINT	Desired width or 0 to keep the original width (in) and actual width (out).
nBitmapHeight	Reference To UDINT	Desired height or 0 to keep the original height (in) and actual height (out).

Return value

HRESULT [▶ 150]

6.1.3.2.2.2 ITcVnBitmapExportNotification

Interface required for sending displayable images as ADS notifications

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnBitmapExportNotification

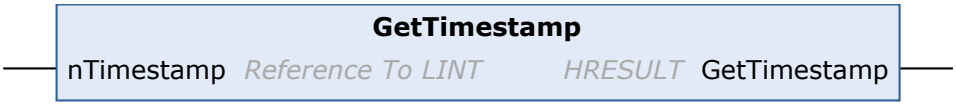
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
GetTimestamp [▶ 412]	ITcVnBitmapExportNotification	Gets the timestamp of the latest image change.
GetBitmapImageRpcUnlocked [▶ 412]	ITcVnBitmapExportNotification	Export the image as a Windows Bitmap (BMP) into a given buffer by means of an unlocked remote procedure call. It lies within the responsibility of the user to ensure that no conflicting accesses can occur.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.2.2.1 GetTimestamp



Gets the timestamp of the latest image change.

Syntax

Definition:

```
METHOD GetTimestamp : HRESULT
VAR_INPUT
    nTimestamp : Reference To LINT;
END_VAR
```

Inputs

Name	Type	Description
nTimestamp	Reference To LINT	Returns the timestamp.

Return value

HRESULT [▶ 150]

6.1.3.2.2.2.2 GetBitmapImageRpcUnlocked

Export the image as a Windows Bitmap (BMP) into a given buffer by means of an unlocked remote procedure call. It lies within the responsibility of the user to ensure that no conflicting accesses can occur.

Syntax

Definition:

```
METHOD GetBitmapImageRpcUnlocked : HRESULT
VAR_INPUT
    nBitmapSize   : Reference To ULINT;
    pDestBuffer   : PVOID;
    nBitmapWidth  : Reference To UDINT;
    nBitmapHeight : Reference To UDINT;
END_VAR
```

Inputs

Name	Type	Description
nBitmapSize	Reference To ULINT	Maximum buffer size (in) and actual buffer size (out).
pDestBuffer	PVOID	Pointer to the destination buffer.
nBitmapWidth	Reference To UDINT	Desired width or 0 to keep the original width (in) and actual width (out).
nBitmapHeight	Reference To UDINT	Desired height or 0 to keep the original height (in) and actual height (out).

Return value

HRESULT [▶ 150]

6.1.3.2.2.3 ITcVnBitmapExportRpcUnlocked

Interface to export an image as a Windows Bitmap (BMP) into a given buffer by means of an unlocked remote procedure call.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnBitmapExport [▶ 409]
    ITcVnBitmapExportRpcUnlocked

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
GetBitmapSize [▶ 410]	ITcVnBitmapExport	Gets size of the image if it is converted to a Windows Bitmap (BMP).
GetBitmapImage [▶ 411]	ITcVnBitmapExport	Export the image as a Windows Bitmap (BMP) into a given buffer.
GetBitmapImageRpcUnlocked [▶ 414]	ITcVnBitmapExportRpcUnlocked	Export the image as a Windows Bitmap (BMP) into a given buffer by means of an unlocked remote procedure call. It lies within the responsibility of the user to ensure that no conflicting accesses can occur.

Weiterführende Informationen

Ein Bitmap kann per Definition nur 8 Bit (für 1-Kanal Bilder) oder 24 Bit (für 3- und 4-Kanal Bilder) haben. Daher werden 16-Bit Bilder sowie Bilder mit Element-Typ REAL oder LREAL für den Bitmap-Export konvertiert. Bei 16-Bit Bildern wird der Wertebereich einfach auf 8 Bit herunterskaliert (0 entspricht 0 und 255 entspricht 65.280). Bei (L)REAL-Bildern wird der Wertebereich [-1, +1] auf [-127, +127] skaliert und als signed Integer übertragen. Werte außerhalb dieser Grenzen werden dabei als -1 bzw. +1 interpretiert.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.2.3.1 GetBitmapImageRpcUnlocked

Export the image as a Windows Bitmap (BMP) into a given buffer by means of an unlocked remote procedure call. It lies within the responsibility of the user to ensure that no conflicting accesses can occur.

Syntax

Definition:

```
METHOD GetBitmapImageRpcUnlocked : HRESULT
VAR_INPUT
    nBitmapSize   : Reference To ULINT;
    pDestBuffer   : PVOID;
    nBitmapWidth  : Reference To UDINT;
    nBitmapHeight : Reference To UDINT;
END_VAR
```

Inputs

Name	Type	Description
nBitmapSize	Reference To ULINT	Maximum buffer size (in) and actual buffer size (out).
pDestBuffer	PVOID	Pointer to the destination buffer.
nBitmapWidth	Reference To UDINT	Desired width or 0 to keep the original width (in) and actual width (out).
nBitmapHeight	Reference To UDINT	Desired height or 0 to keep the original height (in) and actual height (out).

Return value

HRESULT [▶ 150]

6.1.3.2.2.4 ITcVnHistogramExport

Interface for exporting a histogram for an image as an array.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnHistogramExport

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
GetHistogramSize [▶ 415]	ITcVnHistogramExport	Gets the size of the histogram array.
GetHistogramArray [▶ 415]	ITcVnHistogramExport	Gets the histogram array.
GetCustomHistogramSize [▶ 416]	ITcVnHistogramExport	Gets the size of a custom histogram array.
GetCustomHistogramArray [▶ 417]	ITcVnHistogramExport	Gets a custom histogram array.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.2.4.1 GetHistogramSize

Gets the size of the histogram array.

Syntax

Definition:

```
METHOD GetHistogramSize : HRESULT
VAR_INPUT
    nHistogramSize : Reference To ULINT;
END_VAR
```

Inputs

Name	Type	Description
nHistogramSize	Reference To ULINT	Output parameter containing the required buffer size.

Return value

HRESULT [▶ 150]

6.1.3.2.2.4.2 GetHistogramArray

Gets the histogram array.

Syntax

Definition:

```
METHOD GetHistogramArray : HRESULT
VAR_INPUT
    nHistogramSize : Reference To ULINT;
    pDestBuffer    : PVOID;
END_VAR
```

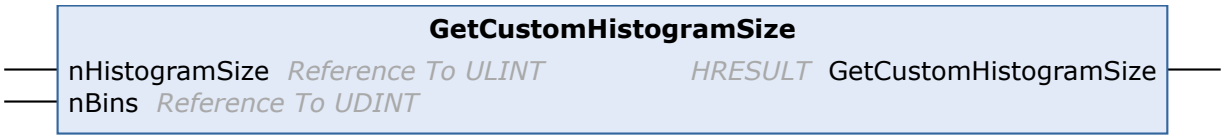
Inputs

Name	Type	Description
nHistogramSize	Reference To ULINT	Maximum buffer size (in) and actual buffer size (out).
pDestBuffer	PVOID	Pointer to the destination buffer.

Return value

HRESULT [▶ 150]

6.1.3.2.2.4.3 GetCustomHistogramSize



Gets the size of a custom histogram array.

Syntax

Definition:

```
METHOD GetCustomHistogramSize : HRESULT
VAR_INPUT
    nHistogramSize : Reference To ULINT;
    nBins          : Reference To UDINT;
END_VAR
```

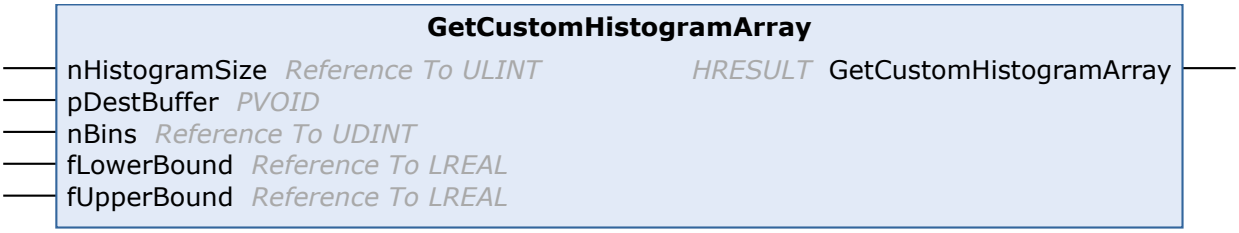
Inputs

Name	Type	Description
nHistogramSize	Reference To ULINT	Output parameter containing the required buffer size.
nBins	Reference To UDINT	Desired number of bins or 0 to keep the default for the corresponding image format (in) and default number of bins (out).

Return value

HRESULT [▶ 150]

6.1.3.2.2.4.4 GetCustomHistogramArray



Gets a custom histogram array.

Syntax

Definition:

```
METHOD GetCustomHistogramArray : HRESULT
VAR_INPUT
    nHistogramSize : Reference To ULINT;
    pDestBuffer    : PVOID;
    nBins          : Reference To UDINT;
    fLowerBound    : Reference To LREAL;
    fUpperBound    : Reference To LREAL;
END_VAR
```

Inputs

Name	Type	Description
nHistogramSize	Reference To ULINT	Maximum buffer size (in) and actual buffer size (out).
pDestBuffer	PVOID	Pointer to the destination buffer.
nBins	Reference To UDINT	Desired number of bins or 0 to keep the default for the corresponding image format (in) and default number of bins (out).
fLowerBound	Reference To LREAL	Lower (inclusive) boundary of the 0-th histogram bin (in), or receive the default if fLowerBound AND fUpperBound are set to 0 (out).
fUpperBound	Reference To LREAL	Upper (exclusive) boundary of the last histogram bin nBins-1 (in), or receive the default if fLowerBound AND fUpperBound are set to 0 (out).

Return value

HRESULT [▶ 150]

6.1.3.2.2.5 ITcVnTiffExport

Interface for exporting an image as tiff.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnTiffExport

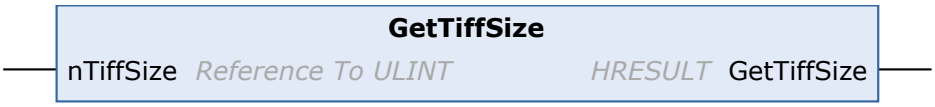
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
GetTiffSize [▶ 418]	ITcVnTiffExport	Gets size of the image if it is converted to tiff.
GetTiffImage [▶ 418]	ITcVnTiffExport	Export the image as tiff into a given buffer.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.2.5.1 GetTiffSize



Gets size of the image if it is converted to tiff.

Syntax

Definition:

```
METHOD GetTiffSize : HRESULT
VAR_INPUT
    nTiffSize : Reference To ULINT;
END_VAR
```

Inputs

Name	Type	Description
nTiffSize	Reference To ULINT	Output parameter containing the required buffer size.

Return value

HRESULT [▶ 150]

6.1.3.2.2.5.2 GetTiffImage

Export the image as tiff into a given buffer.

Syntax

Definition:

```
METHOD GetTiffImage : HRESULT
VAR_INPUT
    nTiffSize   : Reference To ULINT;
    pDestBuffer : PVOID;
END_VAR
```

Inputs

Name	Type	Description
nTiffSize	Reference To ULINT	Maximum buffer size (in) and actual buffer size (out).
pDestBuffer	PVOID	Pointer to the destination buffer.

Return value

HRESULT [▶ 150]

6.1.3.2.2.6 ITcVnTiffExportNotification

Interface for sending displayable images as tiff via ADS notifications.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnTiffExportNotification

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
GetTimestamp [▶ 419]	ITcVnTiffExportNotification	Gets the timestamp of the latest image change.
GetTiffImageRpcUnlocked [▶ 420]	ITcVnTiffExportNotification	Export the image as tiff into a given buffer by means of an unlocked remote procedure call. It lies within the responsibility of the user to ensure that no conflicting accesses can occur.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.2.6.1 GetTimestamp

Gets the timestamp of the latest image change.

Syntax

Definition:

```
METHOD GetTimestamp : HRESULT
VAR_INPUT
    nTimestamp : Reference To LINT;
END_VAR
```

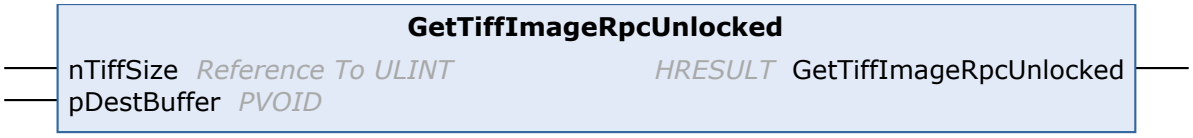
Inputs

Name	Type	Description
nTimestamp	Reference To LINT	Returns the timestamp.

Return value

HRESULT [▶ 150]

6.1.3.2.2.6.2 GetTiffImageRpcUnlocked



Export the image as tiff into a given buffer by means of an unlocked remote procedure call. It lies within the responsibility of the user to ensure that no conflicting accesses can occur.

Syntax

Definition:

```
METHOD GetTiffImageRpcUnlocked : HRESULT
VAR_INPUT
    nTiffSize   : Reference To ULINT;
    pDestBuffer : PVOID;
END_VAR
```

Inputs

Name	Type	Description
nTiffSize	Reference To ULINT	Maximum buffer size (in) and actual buffer size (out).
pDestBuffer	PVOID	Pointer to the destination buffer.

Return value

HRESULT [▶ 150]

6.1.3.2.2.7 ITcVnTiffExportRpcUnlocked

Interface for exporting an image as tiff by means of an unlocked remote procedure call.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnTiffExport [▶ 417]
    ITcVnTiffExportRpcUnlocked

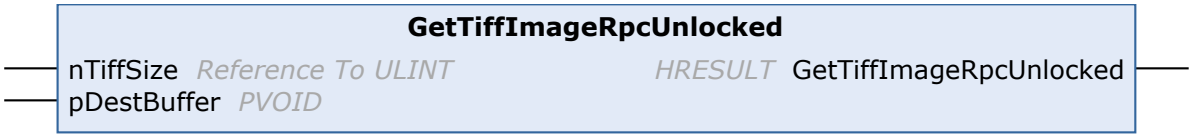
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
GetTiffSize [▶ 418]	ITcVnTiffExport	Gets size of the image if it is converted to tiff.
GetTiffImage [▶ 418]	ITcVnTiffExport	Export the image as tiff into a given buffer.
GetTiffImageRpcUnlocked [▶ 421]	ITcVnTiffExportRpcUnlocked	Export the image as tiff into a given buffer by means of an unlocked remote procedure call. It lies within the responsibility of the user to ensure that no conflicting accesses can occur.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.2.7.1 GetTiffImageRpcUnlocked



Export the image as tiff into a given buffer by means of an unlocked remote procedure call. It lies within the responsibility of the user to ensure that no conflicting accesses can occur.

Syntax

Definition:

```
METHOD GetTiffImageRpcUnlocked : HRESULT
VAR_INPUT
    nTiffSize   : Reference To ULINT;
    pDestBuffer : PVOID;
END_VAR
```

Inputs

Name	Type	Description
nTiffSize	Reference To ULINT	Maximum buffer size (in) and actual buffer size (out).
pDestBuffer	PVOID	Pointer to the destination buffer.

Return value

HRESULT [▶ 150]

6.1.3.2.3 Import

Diese Gruppe enthält Interfaces zur Handhabung des Daten-Imports im Zusammenhang mit Bildern.

6.1.3.2.3.1 ITcVnTiffImport

Interface for importing a tiff image.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnTiffImport

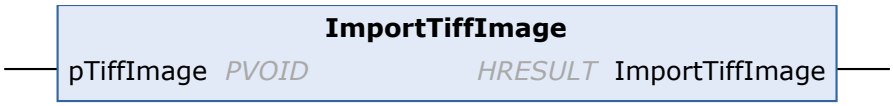
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
ImportTiffImage [▶ 422]	ITcVnTiffImport	Import a tiff image.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.3.1.1 ImportTiffImage



Import a tiff image.

Syntax

Definition:

```
METHOD ImportTiffImage : HRESULT
VAR_INPUT
    pTiffImage : PVOID;
END_VAR
```

Inputs

Name	Type	Description
pTiffImage	PVOID	Pointer to the tiff file.

Return value

HRESULT [▶ 150]

6.1.3.2.4 ITcIoGevImage

Interface for GigE Vision images.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnImageBase [▶ 429]
    ITcIoGevImage

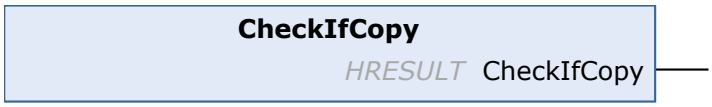
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
GetImageData [▶ 430]	ITcVnImageBase	Gets the data interface (Do not explicitly free the pointer in there!).
GetImageSize [▶ 430]	ITcVnImageBase	Gets the image size.
GetWidth [▶ 431]	ITcVnImageBase	Gets the image width (alternatively use F_VN_GetImageWidth).
GetHeight [▶ 431]	ITcVnImageBase	Gets the image height (alternatively use F_VN_GetImageHeight).
GetXPadding [▶ 432]	ITcVnImageBase	Gets the horizontal padding.
GetYPadding [▶ 432]	ITcVnImageBase	Gets the vertical padding.
GetPixelFormat [▶ 432]	ITcVnImageBase	Gets the pixel format (alternatively use F_VN_GetPixelFormat).
GetImageInfo [▶ 433]	ITcVnImageBase	Gets a struct containing all common meta infos of the image. This basically encompasses all meta information accessible via this interface. (Alternatively use F_VN_GetImageInfo.)
CheckIfCopy [▶ 423]	ITcIoGevImage	Check if image data is a copy.
DecreaseHeight [▶ 424]	ITcIoGevImage	Sets image height to newHeight if it is smaller than the current value.
GetBlockId [▶ 424]	ITcIoGevImage	Gets the GVSP block ID.
GetGevStatus [▶ 424]	ITcIoGevImage	Gets the block status.
GetGvspImageInfo [▶ 425]	ITcIoGevImage	Gets a pointer to the GVSP leader payload.
SetGevStatus [▶ 425]	ITcIoGevImage	Sets the block status.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.4.1 CheckIfCopy



Check if image data is a copy.

Syntax

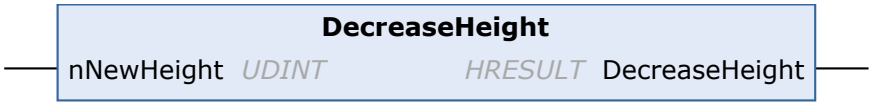
Definition:

```
METHOD CheckIfCopy : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.2.4.2 DecreaseHeight



Sets image height to newHeight if it is smaller than the current value.

Syntax

Definition:

```
METHOD DecreaseHeight : HRESULT
VAR_INPUT
    nNewHeight : UDINT;
END_VAR
```

Inputs

Name	Type	Description
nNewHeight	UDINT	Height value to set.

Return value

HRESULT [▶ 150]

6.1.3.2.4.3 GetBlockId

Gets the GVSP block ID.

Syntax

Definition:

```
METHOD GetBlockId : HRESULT
VAR_INPUT
    nBlockId : Reference To ULINT;
END_VAR
```

Inputs

Name	Type	Description
nBlockId	Reference To ULINT	Returns the block ID.

Return value

HRESULT [▶ 150]

6.1.3.2.4.4 GetGevStatus

Gets the block status.

Syntax

Definition:

```
METHOD GetGevStatus : HRESULT
VAR_INPUT
    nGevStatus : Reference To UINT;
END_VAR
```

Inputs

Name	Type	Description
nGevStatus	Reference To UINT	Returns the GigE Vision status code.

Return value

HRESULT [▶ 150]

6.1.3.2.4.5 GetGvspImageInfo

Gets a pointer to the GVSP leader payload.

Syntax

Definition:

```
METHOD GetGvspImageInfo : HRESULT
VAR_INPUT
    ppGvspImageInfo : Pointer To Pointer To GVSP_IMAGE_INFO;
END_VAR
```

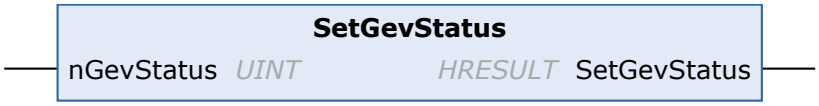
Inputs

Name	Type	Description
ppGvspImageInfo	Pointer To Pointer To GVSP_IMAGE_INFO [▶ 238]	Pointer to the GVSP meta information.

Return value

HRESULT [▶ 150]

6.1.3.2.4.6 SetGevStatus



Sets the block status.

Syntax

Definition:

```
METHOD SetGevStatus : HRESULT
VAR_INPUT
    nGevStatus : UINT;
END_VAR
```

Inputs

Name	Type	Description
nGevStatus	UINT	GigE Vision status code.

Return value

HRESULT [▶ 150]

6.1.3.2.5 ITcVnDisplayableImage

Interface for displayable images.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnDisplayableImage

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.

Weiterführende Informationen

Dieses Interface dient der Anzeige von Bildern über ADS (z. B. mit dem ADS Image Watch). Es ist nicht geeignet, um Bilder zu verarbeiten. Zur Verarbeitung von Bildern wird das Interface ITcVnImage [▶ 426] benötigt. Für weitere Informationen siehe Unterscheidung von verarbeitbaren und anzeigbaren Bildern [▶ 159].

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.6 ITcVnImage

Basic interface for images.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnImageBase [▶ 429]
    ITcVnImage

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
GetImageData [▶ 430]	ITcVnImageBase	Gets the data interface (Do not explicitly free the pointer in there!).
GetImageSize [▶ 430]	ITcVnImageBase	Gets the image size.
GetWidth [▶ 431]	ITcVnImageBase	Gets the image width (alternatively use F_VN_GetImageWidth).
GetHeight [▶ 431]	ITcVnImageBase	Gets the image height (alternatively use F_VN_GetImageHeight).
GetXPadding [▶ 432]	ITcVnImageBase	Gets the horizontal padding.
GetYPadding [▶ 432]	ITcVnImageBase	Gets the vertical padding.
GetPixelFormat [▶ 432]	ITcVnImageBase	Gets the pixel format (alternatively use F_VN_GetPixelFormat).
GetImageInfo [▶ 433]	ITcVnImageBase	Gets a struct containing all common meta infos of the image. This basically encompasses all meta information accessible via this interface. (Alternatively use F_VN_GetImageInfo.)
Init [▶ 427]	ITcVnImage	Initialize an image with an ITcVnImageBase interface.
GetRowPointer [▶ 428]	ITcVnImage	Gets a pointer to a specific row of an image.
ReleaseRowPointer [▶ 429]	ITcVnImage	Release the pointer to a specific row of an image.

Weiterführende Informationen

Das Interface `ITcVnImage` dient zur Handhabung von Bildern [▶ 159]. Es stellt Methoden zum Erlangen von Meta-Informationen wie Bildgröße, Anzahl der Kanäle, Pixeltyp, etc. und zum Zugriff auf die rohen Bilddaten zur Verfügung.

Dieses Interface ist nicht zur Bildanzeige über ADS (z. B. mit dem ADS Image Watch [▶ 141]) geeignet. Um ein Bild über ADS anzuzeigen wird das Interface ITcVnDisplayableImage [▶ 426] benötigt. Eine Konvertierung kann mit folgenden Funktionen vorgenommen werden:

- F_VN_CopyIntoDisplayableImage [▶ 804]
- F_VN_TransformIntoDisplaybleImage [▶ 835]

Verwandte Funktionen

- Basic Image Operations [▶ 795]
- Algebraic Image Operations [▶ 523]

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.6.1 Init

Initialize an image with an ITcVnImageBase interface.

Syntax

Definition:

```
METHOD Init : HRESULT
VAR_INPUT
    ipImageBase : ITcVnImageBase;
END_VAR
```

Inputs

Name	Type	Description
ipImageBase	ITcVnImageBase [▶ 429]	Image from which to obtain the data.

Return value

HRESULT [▶ 150]

6.1.3.2.6.2 GetRowPointer

Gets a pointer to a specific row of an image.

Syntax

Definition:

```
METHOD GetRowPointer : HRESULT
VAR_INPUT
    nRowIndex : UDINT;
    ppRow     : Pointer To PVOID;
END_VAR
```

Inputs

Name	Type	Description
nRowIndex	UDINT	Row index.
ppRow	Pointer To PVOID	Returns a pointer to the requested image row.

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Achten Sie darauf, der zu beschreibende Pointer vom selben elementaren Typ ist wie der Pixel-Typ des Bildes.

HINWEIS

Freigeben eines Row Pointers

Wenn Sie einen Pointer mit GetRowPointer erstellen, müssen Sie diesen unbedingt auch mit der Methode ReleaseRowPointer wieder freigeben!

6.1.3.2.6.3 ReleaseRowPointer

Release the pointer to a specific row of an image.

Syntax

Definition:

```
METHOD ReleaseRowPointer : HRESULT
VAR_INPUT
    ppRow : Pointer To PVOID;
END_VAR
```

Inputs

Name	Type	Description
ppRow	Pointer To PVOID	Pointer to the row pointer to be released.

Return value

HRESULT [▶ 150]

6.1.3.2.7 ITcVnImageBase

Base interface for all image types.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnImageBase

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
GetImageData [▶ 430]	ITcVnImageBase	Gets the data interface (Do not explicitly free the pointer in there!).
GetImageSize [▶ 430]	ITcVnImageBase	Gets the image size.
GetWidth [▶ 431]	ITcVnImageBase	Gets the image width (alternatively use F_VN_GetImageWidth).
GetHeight [▶ 431]	ITcVnImageBase	Gets the image height (alternatively use F_VN_GetImageHeight).
GetXPadding [▶ 432]	ITcVnImageBase	Gets the horizontal padding.
GetYPadding [▶ 432]	ITcVnImageBase	Gets the vertical padding.
GetPixelFormat [▶ 432]	ITcVnImageBase	Gets the pixel format (alternatively use F_VN_GetPixelFormat).
GetImageInfo [▶ 433]	ITcVnImageBase	Gets a struct containing all common meta infos of the image. This basically encompasses all meta information accessible via this interface. (Alternatively use F_VN_GetImageInfo.)

Weiterführende Informationen

Das Interface `ITcVnImageBase` wird für interne Zwecke verwendet und wird vom Anwender im Normalfall nicht benötigt.

Nutzen Sie zum Handhaben von Bildern [▶ 159] stattdessen das Interface ITcVnImage [▶ 426].

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.2.7.1 GetImageData

Gets the data interface (Do not explicitly free the pointer in there!).

Syntax

Definition:

```
METHOD GetImageData : HRESULT
VAR_INPUT
    pipImageData : Pointer To ITcVnData;
END_VAR
```

Inputs

Name	Type	Description
pipImageData	Pointer To ITcVnData [▶ 438]	Returns a pointer to the image data interface.

Return value

HRESULT [▶ 150]

6.1.3.2.7.2 GetImageSize

Gets the image size.

Syntax

Definition:

```
METHOD GetImageSize : HRESULT
VAR_INPUT
    nSize : Reference To ULINT;
END_VAR
```

Inputs

Name	Type	Description
nSize	Reference To ULINT	Returns the image size in bytes.

Return value

HRESULT [▶ 150]

6.1.3.2.7.3 GetWidth

Gets the image width (alternatively use F_VN_GetImageWidth).

Syntax

Definition:

```
METHOD GetWidth : HRESULT
VAR_INPUT
    nWidth : Reference To UDINT;
END_VAR
```

Inputs

Name	Type	Description
nWidth	Reference To UDINT	Returns the image width in pixels.

Return value

HRESULT [▶ 150]

6.1.3.2.7.4 GetHeight

Gets the image height (alternatively use F_VN_GetImageHeight).

Syntax

Definition:

```
METHOD GetHeight : HRESULT
VAR_INPUT
    nHeight : Reference To UDINT;
END_VAR
```

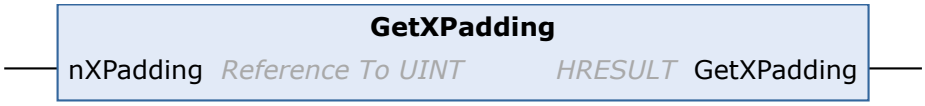
Inputs

Name	Type	Description
nHeight	Reference To UDINT	Returns the image height in pixels.

Return value

HRESULT [▶ 150]

6.1.3.2.7.5 GetXPadding



Gets the horizontal padding.

Syntax

Definition:

```
METHOD GetXPadding : HRESULT
VAR_INPUT
    nXPadding : Reference To UINT;
END_VAR
```

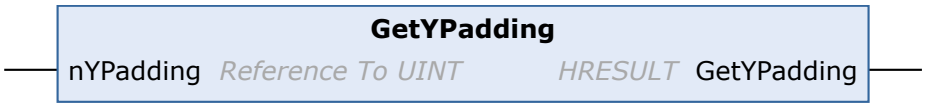
Inputs

Name	Type	Description
nXPadding	Reference To UINT	Returns the horizontal padding in bytes.

Return value

HRESULT [▶ 150]

6.1.3.2.7.6 GetYPadding



Gets the vertical padding.

Syntax

Definition:

```
METHOD GetYPadding : HRESULT
VAR_INPUT
    nYPadding : Reference To UINT;
END_VAR
```

Inputs

Name	Type	Description
nYPadding	Reference To UINT	Returns the vertical padding in bytes.

Return value

HRESULT [▶ 150]

6.1.3.2.7.7 GetPixelFormat

Gets the pixel format (alternatively use F_VN_GetPixelFormat).

Syntax

Definition:

```
METHOD GetPixelFormat : HRESULT
VAR_INPUT
    stPixelFormat : Reference To TcVnPixelFormat;
END_VAR
```

Inputs

Name	Type	Description
stPixelFormat	Reference To TcVnPixelFormat [▶ 256]	Returns a struct describing the pixel format.

Return value

HRESULT [▶ 150]

6.1.3.2.7.8 GetImageInfo

Gets a struct containing all common meta infos of the image. This basically encompasses all meta information accessible via this interface. (Alternatively use F_VN_GetImageInfo.)

Syntax

Definition:

```
METHOD GetImageInfo : HRESULT
VAR_INPUT
    stImageInfo : Reference To TcVnImageInfo;
END_VAR
```

Inputs

Name	Type	Description
stImageInfo	Reference To TcVnImageInfo [▶ 244]	Returns a struct describing the image.

Return value

HRESULT [▶ 150]

6.1.3.3 Machine Learning

Diese Gruppe enthält Interfaces für Maschinelles Lernen.

6.1.3.3.1 ITcVnColorModel

Interface to train and classify an image color.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnMlModel [▶ 435]
    ITcVnColorModel

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
GetTypeGuid [▶ 436]	ITcVnMlModel	Gets the GUID of the model.
GetTypeName [▶ 436]	ITcVnMlModel	Gets the model type name as a string.
TrainImageColor [▶ 434]	ITcVnColorModel	Train the color of the provided image.
ClassifyImageColor [▶ 435]	ITcVnColorModel	Compute the similarity of each image pixel to the trained reference color.

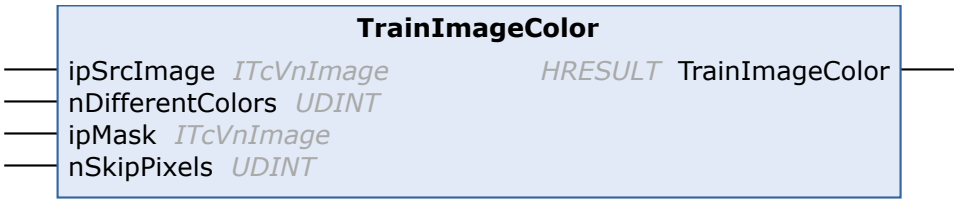
Weiterführende Informationen

Das Interface `ITcVnColorModel` dient zur Beschreibung eines Farbbereichs. Das Modell kann mit der Funktion F_VN_TrainImageColor [▶ 1290] erstellt oder angepasst werden und mit der Funktion F_VN_ReferenceColorSimilarity_ITcVnColorModel [▶ 1283] mit den Farben eines Bildes verglichen werden.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.3.1.1 TrainImageColor



Train the color of the provided image.

Syntax

Definition:

```
METHOD TrainImageColor : HRESULT
VAR_INPUT
    ipSrcImage       : ITcVnImage;
    nDifferentColors : UDINT;
    ipMask           : ITcVnImage;
    nSkipPixels      : UDINT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image.
nDifferentColors	UDINT	Number of different colors to differentiate.
ipMask	ITcVnImage [▶ 426]	Optional image mask.
nSkipPixels	UDINT	Number of pixels to skip between each evaluated color sample (to achieve a better performance). 0 takes every pixel into account and tends to be more accurate.

Return value

HRESULT [▶ 150]

6.1.3.3.1.2 ClassifyImageColor

Compute the similarity of each image pixel to the trained reference color.

Syntax

Definition:

```
METHOD ClassifyImageColor : HRESULT
VAR_INPUT
    ipSrcImage       : ITcVnImage;
    pipDestImage     : Pointer To ITcVnImage;
    fVariance        : REAL;
    fLuminanceWeight : REAL;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image.
pipDestImage	Pointer To ITcVnImage [▶ 426]	Returns the color similarity.
fVariance	REAL	Allowed color variance.
fLuminanceWeight	REAL	Weight the impact of the luminance.

Return value

HRESULT [▶ 150]

6.1.3.3.2 ITcVnMlModel

Interface for a machine learning model.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnMlModel

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
GetTypeGuid [▶ 436]	ITcVnMlModel	Gets the GUID of the model.
GetTypeName [▶ 436]	ITcVnMlModel	Gets the model type name as a string.

Weiterführende Informationen

Das Interface `ITcVnMlModel` dient der Handhabung von Machine Learning Modellen. Zurzeit umfasst dies lediglich die Nutzung des Interfaces ITcVnColorModel [▶ 433].

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.3.2.1 GetTypeGuid

Gets the GUID of the model.

Syntax

Definition:

```
METHOD GetTypeGuid : HRESULT
VAR_INPUT
    nTypeGuid : Reference To GUID;
END_VAR
```

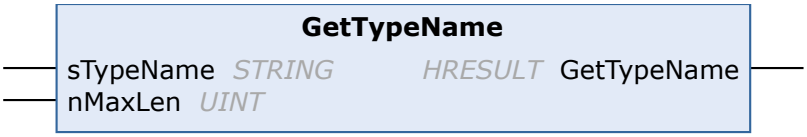
Inputs

Name	Type	Description
nTypeGuid	Reference To GUID	Returns the GUID of the model.

Return value

HRESULT [▶ 150]

6.1.3.3.2.2 GetTypeName



Gets the model type name as a string.

Syntax

Definition:

```
METHOD GetTypeName : HRESULT
VAR_INPUT
    sTypeName : STRING;
    nMaxLen   : UINT;
END_VAR
```

Inputs

Name	Type	Description
sTypeName	STRING	Returns the model type name as a string.
nMaxLen	UINT	Maximum string length allowed to be written in sTypeName.

Return value

HRESULT [▶ 150]

6.1.3.4 Miscellaneous

Diese Gruppe enthält sonstige Interfaces.

6.1.3.4.1 ITcSerializableNotification

Interface required for sending serialized objects as ADS notifications.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcSerializableNotification

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
GetComparisonValue [▶ 437]	ITcSerializableNotification	Gets a value that reflects changes of the underlying object (e.g., the timestamp of the latest object change).
SerializeRpcUnlocked [▶ 438]	ITcSerializableNotification	Serialize an object into a given buffer.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.4.1.1 GetComparisonValue

Gets a value that reflects changes of the underlying object (e.g., the timestamp of the latest object change).

Syntax

Definition:

```
METHOD GetComparisonValue : HRESULT
VAR_INPUT
    nComparisonValue : Reference To LINT;
END_VAR
```

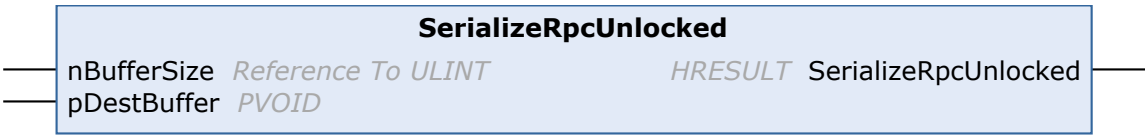
Inputs

Name	Type	Description
nComparisonValue	Reference To LINT	Returns the comparison value.

Return value

HRESULT [▶ 150]

6.1.3.4.1.2 SerializeRpcUnlocked



Serialize an object into a given buffer.

Syntax

Definition:

```
METHOD SerializeRpcUnlocked : HRESULT
VAR_INPUT
    nBufferSize : Reference To ULINT;
    pDestBuffer : PVOID;
END_VAR
```

Inputs

Name	Type	Description
nBufferSize	Reference To ULINT	Maximum buffer size (in) and actually used buffer size (out).
pDestBuffer	PVOID	Pointer to the destination buffer.

Return value

HRESULT [▶ 150]

6.1.3.4.2 ITcVnData

Interface for accessing data.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnData

Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
CheckIfCopy [▶ 439]	ITcVnData	Check if the data are a copy (returns S_OK for copies and S_FALSE otherwise).
CheckIfImage [▶ 439]	ITcVnData	Check if the data belongs to an image (returns S_OK for images and S_FALSE otherwise).
GetSize [▶ 440]	ITcVnData	Gets the size of the data.
CheckDataPointer [▶ 440]	ITcVnData	Check if the data pointer is different from 0 (returns S_OK for non-zero data pointers and S_FALSE otherwise).
GetDataPointer [▶ 440]	ITcVnData	Gets the data pointer (the obtained data pointer must be released by calling ReleaseDataPointer).
ReleaseDataPointer [▶ 441]	ITcVnData	Release the data pointer.

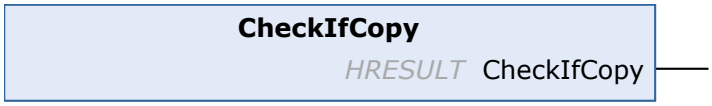
Weiterführende Informationen

Das Interface `ITcVnData` wird für interne Zwecke verwendet und wird vom Anwender im Normalfall nicht benötigt. Nutzen Sie zum Zugriff auf rohe Bilddaten stattdessen die Funktion `GetRowPointer` des Interfaces ITcVnImage [▶ 426].

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.4.2.1 CheckIfCopy



Check if the data are a copy (returns S_OK for copies and S_FALSE otherwise).

Syntax

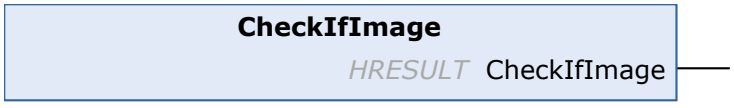
Definition:

```
METHOD CheckIfCopy : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.4.2.2 CheckIfImage



Check if the data belongs to an image (returns S_OK for images and S_FALSE otherwise).

Syntax

Definition:

```
METHOD CheckIfImage : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.4.2.3 GetSize

Gets the size of the data.

Syntax

Definition:

```
METHOD GetSize : HRESULT
VAR_INPUT
    nSize : Reference To ULINT;
END_VAR
```

Inputs

Name	Type	Description
nSize	Reference To ULINT	Size of the stored data in bytes.

Return value

HRESULT [▶ 150]

6.1.3.4.2.4 CheckDataPointer

Check if the data pointer is different from 0 (returns S_OK for non-zero data pointers and S_FALSE otherwise).

Syntax

Definition:

```
METHOD CheckDataPointer : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.4.2.5 GetDataPointer

Gets the data pointer (the obtained data pointer must be released by calling ReleaseDataPointer).

Syntax

Definition:

```
METHOD GetDataPointer : HRESULT
VAR_INPUT
    ppData : Pointer To PVOID;
END_VAR
```

Inputs

Name	Type	Description
ppData	Pointer To PVOID	Returns the data pointer.

Return value

HRESULT [▶ 150]

6.1.3.4.2.6 ReleaseDataPointer

Release the data pointer.

Syntax

Definition:

```
METHOD ReleaseDataPointer : HRESULT
VAR_INPUT
    ppData : Pointer To PVOID;
END_VAR
```

Inputs

Name	Type	Description
ppData	Pointer To PVOID	Pointer to the data pointer to be released.

Return value

HRESULT [▶ 150]

6.1.3.4.3 ITcVnTimestamp

Offers an interface for timestamps.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnTimestamp

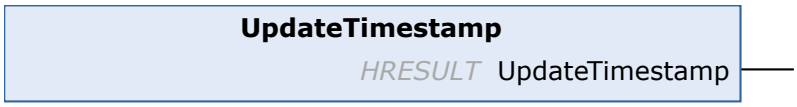
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
UpdateTimestamp [▶ 442]	ITcVnTimestamp	Updates the timestamp to the current time
GetTimestamp [▶ 442]	ITcVnTimestamp	Gets the timestamp

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.4.3.1 UpdateTimestamp



Updates the timestamp to the current time

Syntax

Definition:

```
METHOD UpdateTimestamp : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.3.4.3.2 GetTimestamp

Gets the timestamp

Syntax

Definition:

```
METHOD GetTimestamp : HRESULT
VAR_INPUT
    nTimestamp : Reference To LINT;
END_VAR
```

Inputs

Name	Type	Description
nTimestamp	Reference To LINT	Returns the timestamp

Return value

HRESULT [▶ 150]

6.1.3.4.4 Schnittstelle ITcUnknown

ITcUnknown definiert die Referenzzählung, sowie das Abfragen einer Referenz auf eine spezifischere Schnittstelle.

Syntax

```
TCOM_DECL_INTERFACE("00000001-0000-0000-e000-000000000064", ITcUnknown)
```

Deklariert in: TcInterfaces.h

Benötigtes include: -

Methoden

Name	Beschreibung
TcAddRef [▶ 443]	Inkrementiert den Referenzzähler.
TcQueryInterface [▶ 443]	Abfrage der Referenz an eine implementierte Schnittstelle über der IID.
TcRelease [▶ 444]	Dekrementiert den Referenzzähler.

Jede TcCOM Schnittstelle ist direkt oder indirekt von ITcUnknown abgeleitet. Demzufolge implementiert jede TcCOM Modulklasse ITcUnknown, weil sie von ITComObject abgeleitet ist.

Die standardmäßige Implementierung von ITcUnknown sorgt dafür, dass das Objekt nach Freigabe der letzten Referenz gelöscht wird. Aus diesem Grunde darf ein Schnittstellenzeiger nach dem Aufruf von TcRelease() nicht dereferenziert werden.

6.1.3.4.4.1 Methode ITcUnknown:TcAddRef

Inkrementiert den Referenzzähler und gibt den neuen Wert zurück.

Syntax

```
ULONG TcAddRef( )
```

Rückgabewert

Daraus resultierender Referenzzählwert.

6.1.3.4.4.2 Methode ITcUnknown:TcQueryInterface

Abfrage eines Schnittstellenzeigers in Bezug auf eine Schnittstelle, die per Interface ID (IID) gegeben ist.

Syntax

```
HRESULT TcQueryInterface(RITCID iid, PPVOID pipItf )
```

Rückgabewert

Bei Erfolg wird S_OK („0“) oder ein anderer positiver Wert zurückgegeben, vgl. Rückgabewerte. Erweiterte Meldungen beziehen sich dabei insbesondere auf die Spalte HRESULT in ADS Return Codes [▶ 2966].

Wenn die verlangte Schnittstelle nicht verfügbar ist, gibt die Methode ADSERR_DEVICE_NOINTERFACE zurück.

Parameter

Name	Typ	Beschreibung
iid	RITCID	Schnittstelle IID.
pipItf	PPVOID	Zeiger auf Schnittstellenzeiger. Wird gesetzt, wenn der verlangte Schnittstellentyp von der entsprechenden Instanz verfügbar ist.

Beschreibung

Abfrage der Referenz an eine implementierte Schnittstelle über der IID. Es wird empfohlen, Smart Pointer zu verwenden, um Schnittstellenzeiger zu initialisieren und zu halten.

Variante 1:

```
HRESULT GetTraceLevel(ITcUnkown* ip, TcTraceLevel& tl)
{
HRESULT hr = S_OK;
if (ip != NULL)
{
ITComObjectPtr spObj;
hr = ip->TcQueryInterface(spObj.GetIID(), &spObj);
if (SUCCEEDED(hr))
{
hr = spObj->TcGetObjPara(PID_TcTraceLevel, &tl, sizeof(tl));
}
return hr;
}
}
```

Die mit dem Smart Pointer verbundene Schnittstellen-ID kann in TcQueryInterface als Parameter verwendet werden. Der Operator „&“ wird den Zeiger auf die interne Schnittstellen-Zeiger-Membervariable des Smart Pointers zurückgeben. Variante 1 geht davon aus, dass der Schnittstellenzeiger initialisiert ist, wenn TcQueryInterface Erfolg anzeigt. Wenn der Bereich bleibt, dann gibt der Destructor des Smart Pointers spObj die Referenz frei.

Variante 2:

```
HRESULT GetTraceLevel(ITcUnkown* ip, TcTraceLevel& tl)
{
HRESULT hr = S_OK;
ITComObjectPtr spObj = ip;
if (spObj != NULL)
{
spObj->TcGetObjParam(PID_TcTraceLevel, &tl);
}
else
{
hr = ADS_E_NOINTERFACE;
}
return hr;
}
```

Wenn der Schnittstellenzeiger ip dem Smart Pointer spObj zugewiesen wird, dann wird die TcQueryInterface-Methode implizit aufgerufen mit IID_ITComObject auf der Instanz, auf die ip verweist. Dies führt zu einem kürzeren Code, aber der ursprüngliche Return-Code von TcQueryInterface geht verloren.

6.1.3.4.4.3 Methode ITcUnknown:TcRelease

Diese Methode dekrementiert den Referenzzähler.

Syntax

```
ULONG TcRelease( )
```

Rückgabewert

Daraus resultierender Referenzzählwert.

Beschreibung

Dekrementiert den Referenzzähler und gibt den neuen Wert zurück.

Wenn der Referenzzähler 0 wird, löscht das Objekt sich selbst.

6.1.3.5 Neural Network

6.1.3.5.1 ITcVnNeuralNetwork

Interface for neural networks.

Inheritance Hierarchy

ITcUnknown [▶ 443]
  ITcVnMlModel [▶ 435]
    ITcVnNeuralNetwork

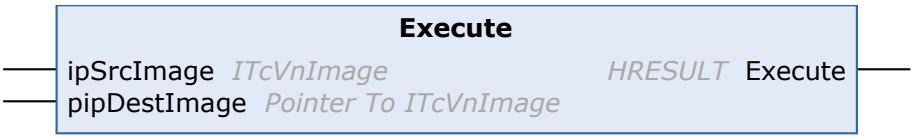
Methods

Name	Origin	Description
TcAddRef [▶ 443]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 443]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 444]	ITcUnknown	Decrements the reference counter.
GetTypeGuid [▶ 436]	ITcVnMlModel	Gets the GUID of the model.
GetTypeName [▶ 436]	ITcVnMlModel	Gets the model type name as a string.
Execute [▶ 445]	ITcVnNeuralNetwork	Execute a neural network using the provided image as input and provide the output of one output layer. The method doesn't perform any pre-processing or transformation.
Execute_SingleLayerOutput [▶ 446]	ITcVnNeuralNetwork	Execute a neural network using the provided image as input and provide the output of the specified output layer. The method doesn't perform any pre-processing or transformation.
Execute_MultiLayerOutput [▶ 447]	ITcVnNeuralNetwork	Execute a neural network using the provided image as input and provide the output of the specified output layers. If sLayerNames is empty, the method returns the output of the unconnected output layers. The method doesn't perform any pre-processing or transformation.
GetLayerNames [▶ 447]	ITcVnNeuralNetwork	Get the name of all layers in the model. If the bOnlyOuput flag is set to true, only the unconnected output layer names are returned.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.3.5.1.1 Execute



Execute a neural network using the provided image as input and provide the output of one output layer. The method doesn't perform any pre-processing or transformation.

Syntax

Definition:

```
METHOD Execute : HRESULT
VAR_INPUT
    ipSrcImage   : ITcVnImage;
    pipDestImage : Pointer To ITcVnImage;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (The image layout must be adapted to match the model's expected input format using e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage functions if necessary.)
pipDestImage	Pointer To ITcVnImage [▶ 426]	Destination image (An appropriate destination image based on the output layer is created. To process the destination image, use e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage if necessary.

Return value

HRESULT [▶ 150]

6.1.3.5.1.2 Execute_SingleLayerOutput

Execute a neural network using the provided image as input and provide the output of the specified output layer. The method doesn't perform any pre-processing or transformation.

Syntax

Definition:

```
METHOD Execute_SingleLayerOutput : HRESULT
VAR_INPUT
    ipSrcImage   : ITcVnImage;
    pipDestImage : Pointer To ITcVnImage;
    sLayerName   : STRING;
END_VAR
```

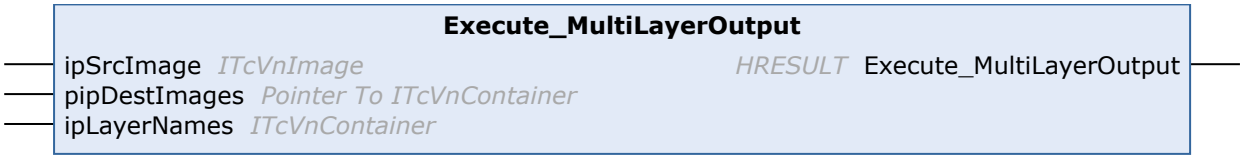
Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (The image layout must be adapted to match the model's expected input format using e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage functions if necessary.)
pipDestImage	Pointer To ITcVnImage [▶ 426]	Destination image (An appropriate destination image based on the output layer is created. To process the destination image, use e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage if necessary.
sLayerName	STRING	Specify the name of the layer for which the output is required. If empty, the output of one output layer is provided.

Return value

HRESULT [▶ 150]

6.1.3.5.1.3 Execute_MultiLayerOutput



Execute a neural network using the provided image as input and provide the output of the specified output layers. If sLayerNames is empty, the method returns the output of the unconnected output layers. The method doesn't perform any pre-processing or transformation.

Syntax

Definition:

```
METHOD Execute_MultiLayerOutput : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    pipDestImages : Pointer To ITcVnContainer;
    ipLayerNames  : ITcVnContainer;
END_VAR
```

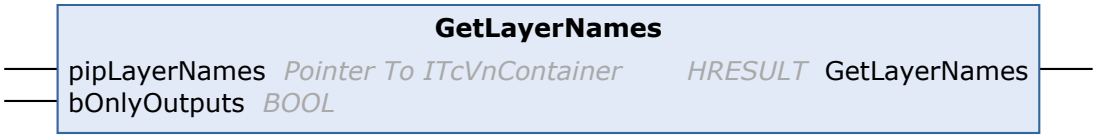
Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (The image layout must be adapted to match the model's expected input format using e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage functions if necessary.)
pipDestImages	Pointer To ITcVnContainer [▶ 385]	Container of images containing the output from the specified layers
ipLayerNames	ITcVnContainer [▶ 385]	Container of layer names whose outputs to be returned (ContainerType_Vector_String_SINT). If set to zero, the container ipDestImages holds the outputs of all the unconnected output layers.

Return value

HRESULT [▶ 150]

6.1.3.5.1.4 GetLayerNames



Get the name of all layers in the model. If the bOnlyOuput flag is set to true, only the unconnected output layer names are returned.

Syntax

Definition:

```
METHOD GetLayerNames : HRESULT
VAR_INPUT
    pipLayerNames : Pointer To ITcVnContainer;
    bOnlyOutputs  : BOOL;
END_VAR
```

Inputs

Name	Type	Description
pipLayerNames	Pointer To ITcVnContainer [▶ 385]	Container with the layers names (ContainerType_Vector_String_SINT)
bOnlyOutputs	BOOL	If true, only the unconnected output layer names are returned

Return value

HRESULT [▶ 150]

6.1.4 Functions

Die Bildverarbeitungsalgorithmen sind in Funktionen gekapselt. Komplexere Algorithmen wurden als Funktionsbaustein umgesetzt und sind unter ImageProcessing [▶ 1703] zu finden.

Aufbau eines Funktionsaufrufs

Funktionen sind wie folgt aufgebaut:

```
hr := F_VN_ProcessImage(<...>, hr);
```

Status variable of type **HRESULT**
Naming convention of vision API functions
Parameters and return values as references

hr	Statusvariable [▶ 150] vom Typ HRESULT. Hiermit kann der Ausführungsstatus jeder Funktion nachvollzogen werden.
F_VN_	Funktionen sind durch das Präfix „F_VN_“ gekennzeichnet.
Parameter <...>	Funktionsparameter erfüllen im Wesentlichen folgenden Zweck: • Eingangsparameter (z. B. Bilder oder Container), die verarbeitet werden sollen. • Parameter zur Einstellung des Algorithmus • Parameter als Referenz zur Rückgabe der Ergebnisse

Gruppierung von Funktionen

Alle Funktionen der TwinCAT Vision Bibliothek sind thematisch in die folgenden Gruppen sortiert:

Bilder

- Basic Image Operations [▶ 795]
- Algebraic Image Operations [▶ 523]
- Image Segmentation [▶ 1356]
- Image Color and Contrast Processing [▶ 1264]
- Image Analysis [▶ 1204]
- Image Filtering [▶ 1304]

Container

- Basic Container Operations [▶ 564]
- Algebraic Container Operations [▶ 480]
- Container Statistics [▶ 891]

Konturen

- Contour Analysis [▶ 985]

Vermessung

- Measurement [▶ 1511]
- Geometric and Coordinate Transformations [▶ 1100]

Code-Qualität

- Code Quality [▶ 843]

Code-Lesung

- Code Reading [▶ 863]

Optische Zeichenerkennung (OCR)

- Optical Character Recognition [▶ 1610]

Zeichnen und Text

- Drawing [▶ 1025]

Machine Learning

- MachineLearning [▶ 1413]
- Neural Network [▶ 1590]

Fortgeschrittene Funktionen

- Fourier Analysis [▶ 1087]
- Keypoint Features [▶ 1367]

Sonstiges

- Control Functions [▶ 1020]
- Miscellaneous [▶ 1559]

Expert-Funktionen

Zu vielen Funktionen existieren Expert-Varianten. Diese enthalten zusätzliche Parameter und sind im Regelfall durch das Namenssuffix „`Exp`“ gekennzeichnet. Bei Funktionen mit Expert-Varianten ist dies zudem unter dem Stichpunkt „Expert-Parameter“ vermerkt.

Die Expert-Variante von F_VN_MaxImage [▶ 547] heißt z. B. F_VN_MaxImagesExp [▶ 548] und enthält den zusätzlichen Parameter `ipMask`.

Laufzeitverhalten

Die Ausführungsdauer einiger Bildverarbeitungsalgorithmen hängt neben ihrer Parametrierung auch vom Bildinhalt ab. Sie ist daher nicht deterministisch. Um dennoch zyklisches Echtzeitverhalten zu gewährleisten, können Watchdogs [▶ 156] auf diese Funktionen oder Funktionsgruppen angewandt werden, die die Funktionen nach einer vorgegebenen Zeit abbrechen.

⚠ WARNUNG

TwinCAT-Absturz bei falscher Verwendung von Funktionen

Bei falscher Verwendung von einigen Funktionen kann es zu Fehlern (und daraus resultierenden TwinCAT-Abstürzen) kommen. Diese werden nicht innerhalb der Funktion abgefangen, da entsprechende Überprüfungen die Ausführungszeit wesentlich verlängern würden. Sie müssen also selbst darauf achten, die Funktionen ordnungsgemäß zu verwenden! Z. B. empfehlen wir, eine Validierung der Eingabeparameter durchzuführen.

6.1.4.1 License Overview

Folgend ist die Zugehörigkeit der TwinCAT Vision Funktionen zu verfügbaren Lizenzen (siehe Lizenzmodell [▶ 12]) aufgelistet. Daraus ist ersichtlich, welche Lizenz für die Anwendung bestimmter Funktionen benötigt wird.

TF7100

Die Basis-Lizenz TC3 Vision Base (TF7100) wird immer benötigt.

TC3 Vision Base

F_VN_AdaptiveThreshold [▶ 1356]

F_VN_AdaptiveThresholdExp [▶ 1358]

F_VN_AddContainers [▶ 509]

F_VN_AddImages [▶ 524]

F_VN_AddImagesWeighted [▶ 525]

F_VN_AddImagesWeightedExp [▶ 526]

F_VN_AddScalarToImage [▶ 527]

F_VN_AddToContainerElements_DINT [▶ 481]

F_VN_AddToContainerElements_INT [▶ 482]

F_VN_AddToContainerElements_LREAL [▶ 482]

F_VN_AddToContainerElements_REAL [▶ 483]

F_VN_AddToContainerElements_SINT [▶ 484]

F_VN_AddToContainerElements_TcVnKeyPoint [▶ 484]

F_VN_AddToContainerElements_TcVnPoint2_DINT [▶ 485]

F_VN_AddToContainerElements_TcVnPoint2_LREAL [▶ 486]

F_VN_AddToContainerElements_TcVnPoint2_REAL [▶ 487]

F_VN_AddToContainerElements_TcVnPoint3_LREAL [▶ 488]

F_VN_AddToContainerElements_TcVnPoint3_REAL [▶ 489]

F_VN_AddToContainerElements_TcVnRectangle_DINT [▶ 489]

F_VN_AddToContainerElements_TcVnRotatedRectangle [▶ 508]

F_VN_AddToContainerElements_TcVnVector2_DINT [▶ 490]

F_VN_AddToContainerElements_TcVnVector2_INT [▶ 491]

F_VN_AddToContainerElements_TcVnVector2_REAL [▶ 492]

F_VN_AddToContainerElements_TcVnVector2_SINT [▶ 493]

F_VN_AddToContainerElements_TcVnVector2_UINT [▶ 494]

F_VN_AddToContainerElements_TcVnVector2_USINT [▶ 494]

F_VN_AddToContainerElements_TcVnVector3_INT [▶ 495]

F_VN_AddToContainerElements_TcVnVector3_REAL [▶ 496]

F_VN_AddToContainerElements_TcVnVector3_SINT [▶ 497]

F_VN_AddToContainerElements_TcVnVector3_UINT [▶ 498]

F_VN_AddToContainerElements_TcVnVector3_USINT [▶ 499]

F_VN_AddToContainerElements_TcVnVector4_DINT [▶ 499]

F_VN_AddToContainerElements_TcVnVector4_INT [▶ 500]

F_VN_AddToContainerElements_TcVnVector4_LREAL [▶ 501]

F_VN_AddToContainerElements_TcVnVector4_REAL [▶ 507]

F_VN_AddToContainerElements_TcVnVector4_SINT [▶ 502]

F_VN_AddToContainerElements_TcVnVector4_UINT [▶ 503]

F_VN_AddToContainerElements_TcVnVector4_USINT [▶ 504]

F_VN_AddToContainerElements_UDINT [▶ 504]

F_VN_AddToContainerElements_UINT [▶ 505]

F_VN_AddToContainerElements_ULINT [▶ 506]

F_VN_AddToContainerElements_USINT [▶ 507]

F_VN_AddVectorToImage [▶ 528]

F_VN_AdjustActiveContour [▶ 1206]

F_VN_AdvanceIterator [▶ 768]

F_VN_AlignRotatedImageRegion [▶ 1142]

F_VN_AlignRotatedImageRegionExp [▶ 1143]

F_VN_AppendToContainer_DINT [▶ 566]

F_VN_AppendToContainer_INT [▶ 566]

F_VN_AppendToContainer_ITcVnContainer [▶ 567]

F_VN_AppendToContainer_ITcVnForwardIterator [▶ 568]

F_VN_AppendToContainer_ITcVnImage [▶ 569]

F_VN_AppendToContainer_LREAL [▶ 569]

F_VN_AppendToContainer_REAL [▶ 570]

F_VN_AppendToContainer_SINT [▶ 571]

F_VN_AppendToContainer_TcVnDMatch [▶ 572]

F_VN_AppendToContainer_TcVnKeyPoint [▶ 572]

F_VN_AppendToContainer_TcVnPoint2_DINT [▶ 573]

F_VN_AppendToContainer_TcVnPoint2_LREAL [▶ 574]

F_VN_AppendToContainer_TcVnPoint2_REAL [▶ 575]

F_VN_AppendToContainer_TcVnPoint3_LREAL [▶ 576]

F_VN_AppendToContainer_TcVnPoint3_REAL [▶ 577]

F_VN_AppendToContainer_TcVnRectangle_DINT [▶ 577]

F_VN_AppendToContainer_TcVnRotatedRectangle [▶ 578]

F_VN_AppendToContainer_TcVnVector2_DINT [▶ 579]

F_VN_AppendToContainer_TcVnVector2_INT [▶ 580]

F_VN_AppendToContainer_TcVnVector2_REAL [▶ 581]

F_VN_AppendToContainer_TcVnVector2_SINT [▶ 582]

F_VN_AppendToContainer_TcVnVector2_UINT [▶ 582]

F_VN_AppendToContainer_TcVnVector2_USINT [▶ 583]

F_VN_AppendToContainer_TcVnVector3_INT [▶ 584]

F_VN_AppendToContainer_TcVnVector3_REAL [▶ 585]

F_VN_AppendToContainer_TcVnVector3_SINT [▶ 586]

F_VN_AppendToContainer_TcVnVector3_UINT [▶ 587]

F_VN_AppendToContainer_TcVnVector3_USINT [▶ 587]

F_VN_AppendToContainer_TcVnVector4_DINT [▶ 588]

F_VN_AppendToContainer_TcVnVector4_INT [▶ 589]

F_VN_AppendToContainer_TcVnVector4_LREAL [▶ 590]

F_VN_AppendToContainer_TcVnVector4_REAL [▶ 591]

F_VN_AppendToContainer_TcVnVector4_SINT [▶ 592]

F_VN_AppendToContainer_TcVnVector4_UINT [▶ 592]

F_VN_AppendToContainer_TcVnVector4_USINT [▶ 593]

F_VN_AppendToContainer_UDINT [▶ 594]

F_VN_AppendToContainer_UINT [▶ 595]

F_VN_AppendToContainer_ULINT [▶ 596]

F_VN_AppendToContainer_USINT [▶ 596]

F_VN_ApplyColorMap [▶ 1265]

F_VN_ApplyLut [▶ 1267]

F_VN_ApplyRotationToAffineTransformation [▶ 1144]

F_VN_ApplyRotationToAffineTransformationExp [▶ 1145]

F_VN_ApplyScalingToAffineTransformation [▶ 1146]

F_VN_ApplyScalingToAffineTransformationExp [▶ 1147]

F_VN_ApplyTranslationToAffineTransformation [▶ 1148]

F_VN_ApplyTranslationToAffineTransformationExp [▶ 1149]

F_VN_ApplyYAxisInversionToAffineTransformation [▶ 1150]

F_VN_ApplyYAxisInversionToAffineTransformationExp [▶ 1151]

F_VN_ApproximatePolygon [▶ 986]

F_VN_BilateralFilter [▶ 1305]

F_VN_BilateralFilterExp [▶ 1309]

F_VN_BitwiseAndContainers [▶ 510]

F_VN_BitwiseAndImages [▶ 529]

F_VN_BitwiseAndImagesExp [▶ 529]

F_VN_BitwiseAndScalarWithImage [▶ 530]

F_VN_BitwiseAndVectorWithImage [▶ 531]

F_VN_BitwiseNotContainer [▶ 511]

F_VN_BitwiseNotImage [▶ 532]

F_VN_BitwiseNotImageExp [▶ 533]

F_VN_BitwiseOrContainers [▶ 511]

F_VN_BitwiseOrImages [▶ 534]

F_VN_BitwiseOrImagesExp [▶ 534]

F_VN_BitwiseOrScalarWithImage [▶ 535]

F_VN_BitwiseOrVectorWithImage [▶ 536]

F_VN_BitwiseXorContainers [▶ 512]

F_VN_BitwiseXorImages [▶ 537]

F_VN_BitwiseXorImagesExp [▶ 538]

F_VN_BitwiseXorScalarWithImage [▶ 539]

F_VN_BitwiseXorVectorWithImage [▶ 539]

F_VN_BlendImages [▶ 540]

F_VN_BoxFilter [▶ 1310]

F_VN_BoxFilterExp [▶ 1311]

F_VN_BrightBorderObjects [▶ 1312]

F_VN_CalibrateCameraPlanar [▶ 1115]

F_VN_CalibrateCameraPlanarExp [▶ 1116]

F_VN_CannyEdgeDetection [▶ 1233]

F_VN_CannyEdgeDetectionExp [▶ 1235]

F_VN_CheckColorRange [▶ 1360]

F_VN_CheckFunctionInitialization [▶ 1560]

F_VN_CheckIfEmpty [▶ 769]

F_VN_CheckIfIteratorIsAtEnd [▶ 770]

F_VN_CheckIfPointIsInsideContour [▶ 987]

F_VN_Clahe [▶ 1268]

F_VN_ClaheExp [▶ 1269]

F_VN_ClipLineToBoundary [▶ 1561]

F_VN_ClipLineToBoundary_ITcVnImage [▶ 1562]

F_VN_ClipLineToBoundary_TcVnRectangle_DINT [▶ 1563]

F_VN_CombineImageChannels [▶ 797]

F_VN_ConnectedComponents [▶ 1236]

F_VN_ConnectedComponentsExp [▶ 1237]

F_VN_ConnectedComponentsWithStats [▶ 1238]

F_VN_ConnectedComponentsWithStatsExp [▶ 1239]

F_VN_ContainerAverage [▶ 973]

F_VN_ContainerAverageElementwise2 [▶ 974]

F_VN_ContainerAverageElementwise3 [▶ 975]

F_VN_ContainerAverageElementwise4 [▶ 976]

F_VN_ContainerAverageVariance [▶ 977]

F_VN_ContainerAverageVarianceElementwise2 [▶ 978]

F_VN_ContainerAverageVarianceElementwise3 [▶ 980]

F_VN_ContainerAverageVarianceElementwise4 [▶ 981]

F_VN_ContourArea [▶ 988]

F_VN_ContourCenterOfMass [▶ 989]

F_VN_ContourCircularity [▶ 990]

F_VN_ContourCircularityExp [▶ 1019]

F_VN_ContourConvexity [▶ 991]

F_VN_ContourEccentricity [▶ 992]

F_VN_ContourElongation [▶ 993]

F_VN_ContourExtremePoint [▶ 994]

F_VN_ContourInertiaRatio [▶ 995]

F_VN_ContourMoments [▶ 996]

F_VN_ContourOrientation [▶ 997]

F_VN_ContourOrientationExp [▶ 998]

F_VN_ContourPerimeter [▶ 999]

F_VN_ContourRoundness [▶ 1000]

F_VN_ConvertCartesianToPolarAngleImage [▶ 1152]

F_VN_ConvertCartesianToPolarAngleImageExp [▶ 1153]

F_VN_ConvertCartesianToPolarAngles [▶ 1153]

F_VN_ConvertCartesianToPolarAnglesExp [▶ 1154]

F_VN_ConvertCartesianToPolarImages [▶ 1155]

F_VN_ConvertCartesianToPolarImagesExp [▶ 1156]

F_VN_ConvertCartesianToPolarMagnitudeImage [▶ 1157]

F_VN_ConvertCartesianToPolarMagnitudes [▶ 1158]

F_VN_ConvertCartesianToPolarPoints [▶ 1159]

F_VN_ConvertCartesianToPolarPointsExp [▶ 1160]

F_VN_ConvertColorSpace [▶ 1270]

F_VN_ConvertContainerType [▶ 771]

F_VN_ConvertDataLayout [▶ 1591]

F_VN_ConvertElementType [▶ 798]

F_VN_ConvertElementTypeExp [▶ 799]

F_VN_ConvertITcUnknownToITcVnBitmapExport [▶ 1564]

F_VN_ConvertITcUnknownToITcVnContainer [▶ 1565]

F_VN_ConvertITcUnknownToITcVnImage [▶ 1566]

F_VN_ConvertITcUnknownToITcVnMlModel [▶ 1567]

F_VN_ConvertPolarToCartesianImages [▶ 1161]

F_VN_ConvertPolarToCartesianImagesExp [▶ 1162]

F_VN_ConvertPolarToCartesianPoints [▶ 1163]

F_VN_ConvertPolarToCartesianPointsExp [▶ 1164]

F_VN_ConvexHullPoints [▶ 1001]

F_VN_ConvexHullPointsExp [▶ 1002]

F_VN_ConvexityDefects [▶ 1003]

F_VN_CopyContainer [▶ 772]

F_VN_CopyContainerElementsConditional_ITcVnContainer [▶ 773]

F_VN_CopyContainerElementsConditional_ITcVnForwardIterator [▶ 774]

F_VN_CopyImage [▶ 799]

F_VN_CopyImageRegion [▶ 801]

F_VN_CopyImageRegionToRegion [▶ 802]

F_VN_CopyIntoDisplayableImage [▶ 804]

F_VN_CountNonZeroPixels [▶ 1241]

F_VN_CreateAssociatedImage [▶ 804]

F_VN_CreateBandpassButterworthFilter [▶ 1088]

F_VN_CreateBandpassGaussianFilter [▶ 1089]

F_VN_CreateBandrejectButterworthFilter [▶ 1090]

F_VN_CreateBandrejectGaussianFilter [▶ 1091]

F_VN_CreateContainer [▶ 775]

F_VN_CreateContainerFromArray [▶ 776]

F_VN_CreateEmptyImage [▶ 805]

F_VN_CreateHighpassButterworthFilter [▶ 1093]

F_VN_CreateHighpassGaussianFilter [▶ 1094]

F_VN_CreateImage [▶ 806]

F_VN_CreateImageAndSetPixels [▶ 807]

F_VN_CreateImageFromArray [▶ 808]

F_VN_CreateLowpassButterworthFilter [▶ 1095]

F_VN_CreateLowpassGaussianFilter [▶ 1096]

F_VN_CreateNDimensionalImage [▶ 837]

F_VN_CreateNDimensionalImageAndSetPixels [▶ 838]

F_VN_CreateNDimensionalImageFromArray [▶ 839]

F_VN_CreateStructuringElement [▶ 1314]

F_VN_CustomElementWiseContainerOperation_ITcVnContainer [▶ 1004]

F_VN_CustomElementWiseContainerOperation_ITcVnForwardIterator [▶ 1005]

F_VN_CustomFilter [▶ 1316]

F_VN_CustomFilterExp [▶ 1319]

F_VN_DarkBorderObjects [▶ 1321]

F_VN_DecomposeAffineTransformation [▶ 1165]

F_VN_DecomposeAffineTransformationExp [▶ 1166]

F_VN_DecomposeHomography [▶ 1167]

F_VN_DecomposeHomographyExp [▶ 1169]

F_VN_DeinitializeFunction [▶ 1568]

F_VN_DetectBlobs [▶ 1207]

F_VN_DetectBlobsExp [▶ 1210]

F_VN_Dft [▶ 1097]

F_VN_DistanceTransformation [▶ 1242]

F_VN_DistanceTransformationExp [▶ 1244]

F_VN_DivideContainers [▶ 513]

F_VN_DivideContainersExp [▶ 514]

F_VN_DivideImageByScalar [▶ 541]

F_VN_DivideImageByVector [▶ 542]

F_VN_DivideImages [▶ 543]

F_VN_DivideScalarByImage [▶ 544]

F_VN_DivideVectorByImage [▶ 545]

F_VN_DoubleThreshold [▶ 1361]

F_VN_DrawArrow [▶ 1026]

F_VN_DrawArrow_TcVnVector4_DINT [▶ 1027]

F_VN_DrawArrowExp [▶ 1028]

F_VN_DrawArrowExp_TcVnVector4_DINT [▶ 1030]

F_VN_DrawCircle [▶ 1031]

F_VN_DrawCircleExp [▶ 1032]

F_VN_DrawCircles [▶ 1033]

F_VN_DrawCirclesExp [▶ 1034]

F_VN_DrawCircularArc [▶ 1035]

F_VN_DrawCircularArcExp [▶ 1036]

F_VN_DrawComponents [▶ 1037]

F_VN_DrawComponentsExp [▶ 1038]

F_VN_DrawContours [▶ 1040]

F_VN_DrawContoursExp [▶ 1041]

F_VN_DrawEllipse [▶ 1043]

F_VN_DrawEllipseExp [▶ 1044]

F_VN_DrawLine [▶ 1047]

F_VN_DrawLine_TcVnVector4_DINT [▶ 1048]

F_VN_DrawLine_TcVnVector4_LREAL [▶ 1050]

F_VN_DrawLineExp [▶ 1051]

F_VN_DrawLineExp_TcVnVector4_DINT [▶ 1052]

F_VN_DrawLineExp_TcVnVector4_LREAL [▶ 1053]

F_VN_DrawLines [▶ 1054]

F_VN_DrawLinesExp [▶ 1056]

F_VN_DrawOrientation [▶ 1059]

F_VN_DrawOrientationExp [▶ 1061]

F_VN_DrawPoint [▶ 1062]

F_VN_DrawPointExp [▶ 1063]

F_VN_DrawPoints [▶ 1064]

F_VN_DrawPointsExp [▶ 1065]

F_VN_DrawPolygon [▶ 1066]

F_VN_DrawPolygonExp [▶ 1067]

F_VN_DrawRectangle [▶ 1068]

F_VN_DrawRectangle_TcVnRectangle_DINT [▶ 1069]

F_VN_DrawRectangle_TcVnRectangle_UDINT [▶ 1070]

F_VN_DrawRotatedRectangle [▶ 1071]

F_VN_DrawRotatedRectangleExp [▶ 1072]

F_VN_ElementwiseExp [▶ 546]

F_VN_ElementwiseLog [▶ 546]

F_VN_EnclosingCircle [▶ 1006]

F_VN_EnclosingRectangle [▶ 1007]

F_VN_EnclosingTriangle [▶ 1008]

F_VN_EraseFromContainer [▶ 777]

F_VN_ExportContainer [▶ 778]

F_VN_ExportContainer_String [▶ 780]

F_VN_ExportContainerSize [▶ 781]

F_VN_ExportImage [▶ 809]

F_VN_ExportImageAsBmp [▶ 810]

F_VN_ExportImageAsBmpExp [▶ 811]

F_VN_ExportImageAsBmpSize [▶ 812]

F_VN_ExportImageAsBmpSizeExp [▶ 813]

F_VN_ExportImageSize [▶ 814]

F_VN_ExportSubContainer [▶ 782]

F_VN_ExportSubContainer_String [▶ 783]

F_VN_ExportSubContainerSize [▶ 784]

F_VN_ExtractContainerRange [▶ 785]

F_VN_FillCircle [▶ 1074]

F_VN_FillContainer_DINT [▶ 597]

F_VN_FillContainer_INT [▶ 598]

F_VN_FillContainer_ITcVnImage [▶ 599]

F_VN_FillContainer_LREAL [▶ 599]

F_VN_FillContainer_REAL [▶ 600]

F_VN_FillContainer_SINT [▶ 601]

F_VN_FillContainer_TcVnDMatch [▶ 601]

F_VN_FillContainer_TcVnKeyPoint [▶ 602]

F_VN_FillContainer_TcVnPoint2_DINT [▶ 603]

F_VN_FillContainer_TcVnPoint2_LREAL [▶ 604]

F_VN_FillContainer_TcVnPoint2_REAL [▶ 605]

F_VN_FillContainer_TcVnPoint3_LREAL [▶ 606]

F_VN_FillContainer_TcVnPoint3_REAL [▶ 606]

F_VN_FillContainer_TcVnRectangle_DINT [▶ 607]

F_VN_FillContainer_TcVnRotatedRectangle [▶ 608]

F_VN_FillContainer_TcVnVector2_DINT [▶ 609]

F_VN_FillContainer_TcVnVector2_INT [▶ 610]

F_VN_FillContainer_TcVnVector2_REAL [▶ 611]

F_VN_FillContainer_TcVnVector2_SINT [▶ 611]

F_VN_FillContainer_TcVnVector2_UINT [▶ 612]

F_VN_FillContainer_TcVnVector2_USINT [▶ 613]

F_VN_FillContainer_TcVnVector3_INT [▶ 614]

F_VN_FillContainer_TcVnVector3_REAL [▶ 615]

F_VN_FillContainer_TcVnVector3_SINT [▶ 616]

F_VN_FillContainer_TcVnVector3_UINT [▶ 617]

F_VN_FillContainer_TcVnVector3_USINT [▶ 617]

F_VN_FillContainer_TcVnVector4_DINT [▶ 618]

F_VN_FillContainer_TcVnVector4_INT [▶ 619]

F_VN_FillContainer_TcVnVector4_LREAL [▶ 620]

F_VN_FillContainer_TcVnVector4_REAL [▶ 621]

F_VN_FillContainer_TcVnVector4_SINT [▶ 622]

F_VN_FillContainer_TcVnVector4_UINT [▶ 622]

F_VN_FillContainer_TcVnVector4_USINT [▶ 623]

F_VN_FillContainer_UDINT [▶ 624]

F_VN_FillContainer_UINT [▶ 625]

F_VN_FillContainer_ULINT [▶ 626]

F_VN_FillContainer_USINT [▶ 626]

F_VN_FillContainerExp_DINT [▶ 627]

F_VN_FillContainerExp_INT [▶ 628]

F_VN_FillContainerExp_ITcVnImage [▶ 629]

F_VN_FillContainerExp_LREAL [▶ 630]

F_VN_FillContainerExp_REAL [▶ 631]

F_VN_FillContainerExp_SINT [▶ 631]

F_VN_FillContainerExp_TcVnDMatch [▶ 632]

F_VN_FillContainerExp_TcVnKeyPoint [▶ 633]

F_VN_FillContainerExp_TcVnPoint2_DINT [▶ 634]

F_VN_FillContainerExp_TcVnPoint2_LREAL [▶ 635]

F_VN_FillContainerExp_TcVnPoint2_REAL [▶ 636]

F_VN_FillContainerExp_TcVnPoint3_LREAL [▶ 637]

F_VN_FillContainerExp_TcVnPoint3_REAL [▶ 638]

F_VN_FillContainerExp_TcVnRectangle_DINT [▶ 639]

F_VN_FillContainerExp_TcVnRotatedRectangle [▶ 640]

F_VN_FillContainerExp_TcVnVector2_DINT [▶ 641]

F_VN_FillContainerExp_TcVnVector2_INT [▶ 642]

F_VN_FillContainerExp_TcVnVector2_REAL [▶ 643]

F_VN_FillContainerExp_TcVnVector2_SINT [▶ 644]

F_VN_FillContainerExp_TcVnVector2_UINT [▶ 645]

F_VN_FillContainerExp_TcVnVector2_USINT [▶ 646]

F_VN_FillContainerExp_TcVnVector3_INT [▶ 647]

F_VN_FillContainerExp_TcVnVector3_REAL [▶ 648]

F_VN_FillContainerExp_TcVnVector3_SINT [▶ 649]

F_VN_FillContainerExp_TcVnVector3_UINT [▶ 650]

F_VN_FillContainerExp_TcVnVector3_USINT [▶ 651]

F_VN_FillContainerExp_TcVnVector4_DINT [▶ 652]

F_VN_FillContainerExp_TcVnVector4_INT [▶ 653]

F_VN_FillContainerExp_TcVnVector4_LREAL [▶ 654]

F_VN_FillContainerExp_TcVnVector4_REAL [▶ 655]

F_VN_FillContainerExp_TcVnVector4_SINT [▶ 656]

F_VN_FillContainerExp_TcVnVector4_UINT [▶ 657]

F_VN_FillContainerExp_TcVnVector4_USINT [▶ 658]

F_VN_FillContainerExp_UDINT [▶ 659]

F_VN_FillContainerExp_UINT [▶ 660]

F_VN_FillContainerExp_ULINT [▶ 661]

F_VN_FillContainerExp_USINT [▶ 662]

F_VN_FillContours [▶ 1075]

F_VN_FillEllipse [▶ 1076]

F_VN_FillHoles [▶ 1322]

F_VN_FillPolygon [▶ 1077]

F_VN_FillRectangle [▶ 1078]

F_VN_FillRotatedRectangle [▶ 1079]

F_VN_FindContourHierarchyExp [▶ 1212]

F_VN_FindContours [▶ 1216]

F_VN_FindContoursExp [▶ 1219]

F_VN_FitEllipse [▶ 1009]

F_VN_FitLine [▶ 1010]

F_VN_FitLineExp [▶ 1011]

F_VN_FlipImage [▶ 1170]

F_VN_FourierDescriptors [▶ 1012]

F_VN_FuseImages [▶ 815]

F_VN_FuseImagesArray [▶ 817]

F_VN_GaussianFilter [▶ 1324]

F_VN_GaussianFilterExp [▶ 1327]

F_VN_GenerateAffineTransformationUnitMatrix2D [▶ 1171]

F_VN_GenerateColorMap [▶ 1270]

F_VN_GenerateCustomColorMap [▶ 1272]

F_VN_GetAffineTransformation [▶ 1172]

F_VN_GetAffineTransformation2D [▶ 1173]

F_VN_GetAffineTransformation2DExp [▶ 1174]

F_VN_GetAt_DINT [▶ 663]

F_VN_GetAt_INT [▶ 664]

F_VN_GetAt_ITcVnContainer [▶ 665]

F_VN_GetAt_ITcVnImage [▶ 665]

F_VN_GetAt_LREAL [▶ 666]

F_VN_GetAt_REAL [▶ 667]

F_VN_GetAt_SINT [▶ 668]

F_VN_GetAt_TcVnDMatch [▶ 669]

F_VN_GetAt_TcVnKeyPoint [▶ 670]

F_VN_GetAt_TcVnPoint2_DINT [▶ 671]

F_VN_GetAt_TcVnPoint2_LREAL [▶ 672]

F_VN_GetAt_TcVnPoint2_REAL [▶ 673]

F_VN_GetAt_TcVnPoint3_LREAL [▶ 674]

F_VN_GetAt_TcVnPoint3_REAL [▶ 675]

F_VN_GetAt_TcVnRectangle_DINT [▶ 676]

F_VN_GetAt_TcVnRotatedRectangle [▶ 677]

F_VN_GetAt_TcVnVector2_DINT [▶ 678]

F_VN_GetAt_TcVnVector2_INT [▶ 679]

F_VN_GetAt_TcVnVector2_REAL [▶ 680]

F_VN_GetAt_TcVnVector2_SINT [▶ 681]

F_VN_GetAt_TcVnVector2_UINT [▶ 682]

F_VN_GetAt_TcVnVector2_USINT [▶ 683]

F_VN_GetAt_TcVnVector3_INT [▶ 684]

F_VN_GetAt_TcVnVector3_REAL [▶ 685]

F_VN_GetAt_TcVnVector3_SINT [▶ 686]

F_VN_GetAt_TcVnVector3_UINT [▶ 687]

F_VN_GetAt_TcVnVector3_USINT [▶ 688]

F_VN_GetAt_TcVnVector4_DINT [▶ 689]

F_VN_GetAt_TcVnVector4_INT [▶ 690]

F_VN_GetAt_TcVnVector4_LREAL [▶ 691]

F_VN_GetAt_TcVnVector4_REAL [▶ 692]

F_VN_GetAt_TcVnVector4_SINT [▶ 693]

F_VN_GetAt_TcVnVector4_UINT [▶ 694]

F_VN_GetAt_TcVnVector4_USINT [▶ 695]

F_VN_GetAt_UDINT [▶ 696]

F_VN_GetAt_UINT [▶ 697]

F_VN_GetAt_ULINT [▶ 698]

F_VN_GetAt_USINT [▶ 699]

F_VN_GetConnectedComponent [▶ 1245]

F_VN_GetConnectedComponentExp [▶ 1246]

F_VN_GetContainer [▶ 786]

F_VN_GetContainerExp [▶ 787]

F_VN_GetForwardIterator [▶ 788]

F_VN_GetImageChannel [▶ 818]

F_VN_GetImageHeight [▶ 819]

F_VN_GetImageInfo [▶ 820]

F_VN_GetImageWidth [▶ 820]

F_VN_GetLayoutInfo [▶ 840]

F_VN_GetNumberOfElements [▶ 789]

F_VN_GetPerspectiveTransformation [▶ 1175]

F_VN_GetPixel [▶ 821]

F_VN_GetPixelFormat [▶ 822]

F_VN_GetRandomAccessIterator [▶ 790]

F_VN_GetRoi [▶ 823]

F_VN_GetTimestamp [▶ 1568]

F_VN_Histogram [▶ 1273]

F_VN_HistogramEqualization [▶ 1274]

F_VN_HistogramEqualizationExp [▶ 1275]

F_VN_HistogramExp [▶ 1276]

F_VN_Homography [▶ 1176]

F_VN_HomographyExp [▶ 1177]

F_VN_HoughCircles [▶ 1221]

F_VN_HoughCirclesExp [▶ 1222]

F_VN_HoughLines [▶ 1223]

F_VN_HoughLinesExp [▶ 1224]

F_VN_HoughLinesP [▶ 1225]

F_VN_HoughLinesPExp [▶ 1226]

F_VN_HuMomentInvariants [▶ 1569]

F_VN_ImageAbsoluteAverage [▶ 1262]

F_VN_ImageAverage [▶ 1247]

F_VN_ImageAverageExp [▶ 1247]

F_VN_ImageAverageStdDev [▶ 1248]

F_VN_ImageAverageStdDevExp [▶ 1249]

F_VN_ImageCenterOfMass [▶ 1250]

F_VN_ImageCenterOfMassExp [▶ 1252]

F_VN_ImageMedian [▶ 1253]

F_VN_ImageMedianExp [▶ 1254]

F_VN_ImageMoments [▶ 1255]

F_VN_IncrementIterator [▶ 791]

F_VN_InitMatrixStruct [▶ 1570]

F_VN_InsertIntoContainer_DINT [▶ 700]

F_VN_InsertIntoContainer_INT [▶ 700]

F_VN_InsertIntoContainer_ITcVnContainer [▶ 701]

F_VN_InsertIntoContainer_ITcVnForwardIterator [▶ 702]

F_VN_InsertIntoContainer_ITcVnImage [▶ 703]

F_VN_InsertIntoContainer_LREAL [▶ 704]

F_VN_InsertIntoContainer_REAL [▶ 704]

F_VN_InsertIntoContainer_SINT [▶ 705]

F_VN_InsertIntoContainer_TcVnDMatch [▶ 706]

F_VN_InsertIntoContainer_TcVnKeyPoint [▶ 707]

F_VN_InsertIntoContainer_TcVnPoint2_DINT [▶ 708]

F_VN_InsertIntoContainer_TcVnPoint2_LREAL [▶ 709]

F_VN_InsertIntoContainer_TcVnPoint2_REAL [▶ 710]

F_VN_InsertIntoContainer_TcVnPoint3_LREAL [▶ 710]

F_VN_InsertIntoContainer_TcVnPoint3_REAL [▶ 711]

F_VN_InsertIntoContainer_TcVnRectangle_DINT [▶ 712]

F_VN_InsertIntoContainer_TcVnRotatedRectangle [▶ 732]

F_VN_InsertIntoContainer_TcVnVector2_DINT [▶ 713]

F_VN_InsertIntoContainer_TcVnVector2_INT [▶ 714]

F_VN_InsertIntoContainer_TcVnVector2_REAL [▶ 715]

F_VN_InsertIntoContainer_TcVnVector2_SINT [▶ 716]

F_VN_InsertIntoContainer_TcVnVector2_UINT [▶ 716]

F_VN_InsertIntoContainer_TcVnVector2_USINT [▶ 717]

F_VN_InsertIntoContainer_TcVnVector3_INT [▶ 718]

F_VN_InsertIntoContainer_TcVnVector3_REAL [▶ 719]

F_VN_InsertIntoContainer_TcVnVector3_SINT [▶ 720]

F_VN_InsertIntoContainer_TcVnVector3_UINT [▶ 721]

F_VN_InsertIntoContainer_TcVnVector3_USINT [▶ 722]

F_VN_InsertIntoContainer_TcVnVector4_DINT [▶ 722]

F_VN_InsertIntoContainer_TcVnVector4_INT [▶ 723]

F_VN_InsertIntoContainer_TcVnVector4_LREAL [▶ 724]

F_VN_InsertIntoContainer_TcVnVector4_REAL [▶ 731]

F_VN_InsertIntoContainer_TcVnVector4_SINT [▶ 725]

F_VN_InsertIntoContainer_TcVnVector4_UINT [▶ 726]

F_VN_InsertIntoContainer_TcVnVector4_USINT [▶ 727]

F_VN_InsertIntoContainer_UDINT [▶ 728]

F_VN_InsertIntoContainer_UINT [▶ 728]

F_VN_InsertIntoContainer_ULINT [▶ 729]

F_VN_InsertIntoContainer_USINT [▶ 730]

F_VN_InverseDft [▶ 1098]

F_VN_InvertAffineTransformation [▶ 1178]

F_VN_InvertImageColor [▶ 1278]

F_VN_InvertImageColorExp [▶ 1279]

F_VN_InvertMatrix3x3 [▶ 1572]

F_VN_IteratorDistance [▶ 792]

F_VN_LaplacianFilter [▶ 1328]

F_VN_LaplacianFilterExp [▶ 1331]

F_VN_LineIntersectionPoint [▶ 1573]

F_VN_LineIntersectionPointAndAngle [▶ 1574]

F_VN_LocalMaxima [▶ 1333]

F_VN_LocalMinima [▶ 1334]

F_VN_MaxContainer [▶ 515]

F_VN_MaxElement [▶ 982]

F_VN_MaxElementElementwise_DINT [▶ 892]

F_VN_MaxElementElementwise_INT [▶ 892]

F_VN_MaxElementElementwise_LREAL [▶ 893]

F_VN_MaxElementElementwise_REAL [▶ 894]

F_VN_MaxElementElementwise_SINT [▶ 895]

F_VN_MaxElementElementwise_TcVnPoint2_DINT [▶ 896]

F_VN_MaxElementElementwise_TcVnPoint2_LREAL [▶ 897]

F_VN_MaxElementElementwise_TcVnPoint2_REAL [▶ 898]

F_VN_MaxElementElementwise_TcVnPoint3_LREAL [▶ 898]

F_VN_MaxElementElementwise_TcVnPoint3_REAL [▶ 899]

F_VN_MaxElementElementwise_TcVnVector2_DINT [▶ 900]

F_VN_MaxElementElementwise_TcVnVector2_INT [▶ 901]

F_VN_MaxElementElementwise_TcVnVector2_REAL [▶ 902]

F_VN_MaxElementElementwise_TcVnVector2_SINT [▶ 903]

F_VN_MaxElementElementwise_TcVnVector2_UINT [▶ 903]

F_VN_MaxElementElementwise_TcVnVector2_USINT [▶ 904]

F_VN_MaxElementElementwise_TcVnVector3_INT [▶ 905]

F_VN_MaxElementElementwise_TcVnVector3_REAL [▶ 906]

F_VN_MaxElementElementwise_TcVnVector3_SINT [▶ 907]

F_VN_MaxElementElementwise_TcVnVector3_UINT [▶ 908]

F_VN_MaxElementElementwise_TcVnVector3_USINT [▶ 908]

F_VN_MaxElementElementwise_TcVnVector4_DINT [▶ 909]

F_VN_MaxElementElementwise_TcVnVector4_INT [▶ 910]

F_VN_MaxElementElementwise_TcVnVector4_LREAL [▶ 911]

F_VN_MaxElementElementwise_TcVnVector4_REAL [▶ 918]

F_VN_MaxElementElementwise_TcVnVector4_SINT [▶ 912]

F_VN_MaxElementElementwise_TcVnVector4_UINT [▶ 913]

F_VN_MaxElementElementwise_TcVnVector4_USINT [▶ 913]

F_VN_MaxElementElementwise_UDINT [▶ 914]

F_VN_MaxElementElementwise_UINT [▶ 915]

F_VN_MaxElementElementwise_ULINT [▶ 916]

F_VN_MaxElementElementwise_USINT [▶ 917]

F_VN_MaxImage [▶ 547]

F_VN_MaxImageExp [▶ 548]

F_VN_MaxImageWithScalar [▶ 549]

F_VN_MaxImageWithScalarExp [▶ 550]

F_VN_MaxImageWithVector [▶ 551]

F_VN_MaxImageWithVectorExp [▶ 552]

F_VN_MaxPixelValue [▶ 1256]

F_VN_MaxPixelValueExp [▶ 1257]

F_VN_MedianElement [▶ 983]

F_VN_MedianElementElementwise_DINT [▶ 919]

F_VN_MedianElementElementwise_INT [▶ 920]

F_VN_MedianElementElementwise_LREAL [▶ 920]

F_VN_MedianElementElementwise_REAL [▶ 921]

F_VN_MedianElementElementwise_SINT [▶ 922]

F_VN_MedianElementElementwise_TcVnPoint2_DINT [▶ 923]

F_VN_MedianElementElementwise_TcVnPoint2_LREAL [▶ 924]

F_VN_MedianElementElementwise_TcVnPoint2_REAL [▶ 925]

F_VN_MedianElementElementwise_TcVnPoint3_LREAL [▶ 925]

F_VN_MedianElementElementwise_TcVnPoint3_REAL [▶ 926]

F_VN_MedianElementElementwise_TcVnVector2_DINT [▶ 927]

F_VN_MedianElementElementwise_TcVnVector2_INT [▶ 928]

F_VN_MedianElementElementwise_TcVnVector2_REAL [▶ 929]

F_VN_MedianElementElementwise_TcVnVector2_SINT [▶ 930]

F_VN_MedianElementElementwise_TcVnVector2_UINT [▶ 930]

F_VN_MedianElementElementwise_TcVnVector2_USINT [▶ 931]

F_VN_MedianElementElementwise_TcVnVector3_INT [▶ 932]

F_VN_MedianElementElementwise_TcVnVector3_REAL [▶ 933]

F_VN_MedianElementElementwise_TcVnVector3_SINT [▶ 934]

F_VN_MedianElementElementwise_TcVnVector3_UINT [▶ 935]

F_VN_MedianElementElementwise_TcVnVector3_USINT [▶ 935]

F_VN_MedianElementElementwise_TcVnVector4_DINT [▶ 936]

F_VN_MedianElementElementwise_TcVnVector4_INT [▶ 937]

F_VN_MedianElementElementwise_TcVnVector4_LREAL [▶ 938]

F_VN_MedianElementElementwise_TcVnVector4_REAL [▶ 945]

F_VN_MedianElementElementwise_TcVnVector4_SINT [▶ 939]

F_VN_MedianElementElementwise_TcVnVector4_UINT [▶ 940]

F_VN_MedianElementElementwise_TcVnVector4_USINT [▶ 940]

F_VN_MedianElementElementwise_UDINT [▶ 941]

F_VN_MedianElementElementwise_UINT [▶ 942]

F_VN_MedianElementElementwise_ULINT [▶ 943]

F_VN_MedianElementElementwise_USINT [▶ 944]

F_VN_MedianFilter [▶ 1335]

F_VN_MinContainer [▶ 516]

F_VN_MinElement [▶ 984]

F_VN_MinElementElementwise_DINT [▶ 946]

F_VN_MinElementElementwise_INT [▶ 946]

F_VN_MinElementElementwise_LREAL [▶ 947]

F_VN_MinElementElementwise_REAL [▶ 948]

F_VN_MinElementElementwise_SINT [▶ 949]

F_VN_MinElementElementwise_TcVnPoint2_DINT [▶ 950]

F_VN_MinElementElementwise_TcVnPoint2_LREAL [▶ 951]

F_VN_MinElementElementwise_TcVnPoint2_REAL [▶ 952]

F_VN_MinElementElementwise_TcVnPoint3_LREAL [▶ 952]

F_VN_MinElementElementwise_TcVnPoint3_REAL [▶ 953]

F_VN_MinElementElementwise_TcVnVector2_DINT [▶ 954]

F_VN_MinElementElementwise_TcVnVector2_INT [▶ 955]

F_VN_MinElementElementwise_TcVnVector2_REAL [▶ 956]

F_VN_MinElementElementwise_TcVnVector2_SINT [▶ 957]

F_VN_MinElementElementwise_TcVnVector2_UINT [▶ 957]

F_VN_MinElementElementwise_TcVnVector2_USINT [▶ 958]

F_VN_MinElementElementwise_TcVnVector3_INT [▶ 959]

F_VN_MinElementElementwise_TcVnVector3_REAL [▶ 960]

F_VN_MinElementElementwise_TcVnVector3_SINT [▶ 961]

F_VN_MinElementElementwise_TcVnVector3_UINT [▶ 962]

F_VN_MinElementElementwise_TcVnVector3_USINT [▶ 962]

F_VN_MinElementElementwise_TcVnVector4_DINT [▶ 963]

F_VN_MinElementElementwise_TcVnVector4_INT [▶ 964]

F_VN_MinElementElementwise_TcVnVector4_LREAL [▶ 965]

F_VN_MinElementElementwise_TcVnVector4_REAL [▶ 972]

F_VN_MinElementElementwise_TcVnVector4_SINT [▶ 966]

F_VN_MinElementElementwise_TcVnVector4_UINT [▶ 967]

F_VN_MinElementElementwise_TcVnVector4_USINT [▶ 967]

F_VN_MinElementElementwise_UDINT [▶ 968]

F_VN_MinElementElementwise_UINT [▶ 969]

F_VN_MinElementElementwise_ULINT [▶ 970]

F_VN_MinElementElementwise_USINT [▶ 971]

F_VN_MinImage [▶ 553]

F_VN_MinImageExp [▶ 553]

F_VN_MinImageWithScalar [▶ 554]

F_VN_MinImageWithScalarExp [▶ 555]

F_VN_MinImageWithVector [▶ 556]

F_VN_MinImageWithVectorExp [▶ 557]

F_VN_MinPixelValue [▶ 1258]

F_VN_MinPixelValueExp [▶ 1259]

F_VN_MixImageChannels [▶ 824]

F_VN_MorphologicalOperator [▶ 1338]

F_VN_MultiplyContainers [▶ 517]

F_VN_MultiplyImages [▶ 558]

F_VN_MultiplyImageWithScalar [▶ 559]

F_VN_MultiplyImageWithVector [▶ 559]

F_VN_MultiplyMatrices [▶ 1575]

F_VN_MultiplyWithContainerElements1 [▶ 518]

F_VN_MultiplyWithContainerElements2 [▶ 519]

F_VN_MultiplyWithContainerElements3 [▶ 519]

F_VN_MultiplyWithContainerElements4 [▶ 520]

F_VN_NegateContainer [▶ 521]

F_VN_NonMaxSuppression [▶ 1602]

F_VN_NonMaxSuppressionExp [▶ 1603]

F_VN_NormalizeImage [▶ 1280]

F_VN_NormalizeImageExp [▶ 1281]

F_VN_NormalizeImageForDisplay [▶ 1282]

F_VN_OptimalDftSize [▶ 1099]

F_VN_OrthogonalVector_TcVnVector2_DINT [▶ 1576]

F_VN_OrthogonalVector_TcVnVector2_LREAL [▶ 1576]

F_VN_OrthogonalVector_TcVnVector2_REAL [▶ 1577]

F_VN_PadImageBorder [▶ 841]

F_VN_PadImageBorderExp [▶ 842]

F_VN_PerspectiveTransformation [▶ 1179]

F_VN_PlotIntensityProfile [▶ 1080]

F_VN_PlotIntensityProfileExp [▶ 1081]

F_VN_PointToLineDistance_TcVnPoint2_DINT [▶ 1578]

F_VN_PointToLineDistance_TcVnPoint2_LREAL [▶ 1579]

F_VN_PointToLineDistance_TcVnPoint2_REAL [▶ 1580]

F_VN_PointToPointDirection_TcVnPoint2_DINT [▶ 1581]

F_VN_PointToPointDirection_TcVnPoint2_LREAL [▶ 1582]

F_VN_PointToPointDirection_TcVnPoint2_REAL [▶ 1583]

F_VN_PointToPointDistance_TcVnPoint2_DINT [▶ 1584]

F_VN_PointToPointDistance_TcVnPoint2_LREAL [▶ 1585]

F_VN_PointToPointDistance_TcVnPoint2_REAL [▶ 1586]

F_VN_PutLabel [▶ 1082]

F_VN_PutLabelExp [▶ 1083]

F_VN_PutText [▶ 1084]

F_VN_PutTextExp [▶ 1086]

F_VN_PyramidDown [▶ 1180]

F_VN_PyramidUp [▶ 1181]

F_VN_Reduce [▶ 1604]

F_VN_ReduceArg [▶ 1605]

F_VN_ReferenceColorSimilarity_ITcVnColorModel [▶ 1283]

F_VN_ReferenceColorSimilarity_ITcVnMlModel [▶ 1284]

F_VN_ReferenceColorSimilarity_TcVnVector3_LREAL [▶ 1285]

F_VN_ReferenceColorSimilarityExp_ITcVnColorModel [▶ 1286]

F_VN_ReferenceColorSimilarityExp_ITcVnMlModel [▶ 1287]

F_VN_ReferenceColorSimilarityExp_TcVnVector3_LREAL [▶ 1288]

F_VN_RegionOrientation [▶ 1260]

F_VN_RegionOrientationExp [▶ 1261]

F_VN_ReinterpretUnsupportedImage [▶ 843]

F_VN_RemapImageToLogPolarSpace [▶ 1182]

F_VN_RemapImageToLogPolarSpaceExp [▶ 1183]

F_VN_RemapImageToLogPolarSpaceExp2 [▶ 1184]

F_VN_RemapImageToPolarSpace [▶ 1185]

F_VN_RemapImageToPolarSpaceExp [▶ 1186]

F_VN_RemapImageToPolarSpaceExp2 [▶ 1187]

F_VN_RemoveLocalMaxima [▶ 1339]

F_VN_RemoveLocalMinima [▶ 1341]

F_VN_ReserveContainerMemory [▶ 793]

F_VN_ResetRoi [▶ 825]

F_VN_ReshapeImage [▶ 1606]

F_VN_ResizeImage [▶ 1188]

F_VN_ResizeImageExp [▶ 1189]

F_VN_ReverseContainer [▶ 793]

F_VN_RotatedRectangleCorners [▶ 1587]

F_VN_RotatedRectangleIntersection [▶ 1587]

F_VN_RotateImage [▶ 1191]

F_VN_RotateImageExp [▶ 1191]

F_VN_RotateImageExp2 [▶ 1192]

F_VN_SauvolaThreshold [▶ 1366]

F_VN_ScharrFilter [▶ 1342]

F_VN_ScharrFilterExp [▶ 1344]

F_VN_SeparableCustomFilter [▶ 1346]

F_VN_SeparableCustomFilterExp [▶ 1348]

F_VN_SetAt_DINT [▶ 733]

F_VN_SetAt_INT [▶ 733]

F_VN_SetAt_ITcVnContainer [▶ 734]

F_VN_SetAt_ITcVnImage [▶ 735]

F_VN_SetAt_LREAL [▶ 736]

F_VN_SetAt_REAL [▶ 737]

F_VN_SetAt_SINT [▶ 737]

F_VN_SetAt_TcVnDMatch [▶ 738]

F_VN_SetAt_TcVnKeyPoint [▶ 739]

F_VN_SetAt_TcVnPoint2_DINT [▶ 740]

F_VN_SetAt_TcVnPoint2_LREAL [▶ 741]

F_VN_SetAt_TcVnPoint2_REAL [▶ 742]

F_VN_SetAt_TcVnPoint3_LREAL [▶ 743]

F_VN_SetAt_TcVnPoint3_REAL [▶ 744]

F_VN_SetAt_TcVnRectangle_DINT [▶ 745]

F_VN_SetAt_TcVnRotatedRectangle [▶ 746]

F_VN_SetAt_TcVnVector2_DINT [▶ 747]

F_VN_SetAt_TcVnVector2_INT [▶ 748]

F_VN_SetAt_TcVnVector2_REAL [▶ 749]

F_VN_SetAt_TcVnVector2_SINT [▶ 750]

F_VN_SetAt_TcVnVector2_UINT [▶ 751]

F_VN_SetAt_TcVnVector2_USINT [▶ 752]

F_VN_SetAt_TcVnVector3_INT [▶ 753]

F_VN_SetAt_TcVnVector3_REAL [▶ 754]

F_VN_SetAt_TcVnVector3_SINT [▶ 755]

F_VN_SetAt_TcVnVector3_UINT [▶ 756]

F_VN_SetAt_TcVnVector3_USINT [▶ 757]

F_VN_SetAt_TcVnVector4_DINT [▶ 758]

F_VN_SetAt_TcVnVector4_INT [▶ 759]

F_VN_SetAt_TcVnVector4_LREAL [▶ 760]

F_VN_SetAt_TcVnVector4_REAL [▶ 761]

F_VN_SetAt_TcVnVector4_SINT [▶ 762]

F_VN_SetAt_TcVnVector4_UINT [▶ 763]

F_VN_SetAt_TcVnVector4_USINT [▶ 764]

F_VN_SetAt_UDINT [▶ 765]

F_VN_SetAt_UINT [▶ 766]

F_VN_SetAt_ULINT [▶ 766]

F_VN_SetAt_USINT [▶ 767]

F_VN_SetContainer [▶ 794]

F_VN_SetImageChannel [▶ 826]

F_VN_SetIteratorToBegin [▶ 795]

F_VN_SetPixel [▶ 827]

F_VN_SetPixels [▶ 828]

F_VN_SetPixelsExp [▶ 829]

F_VN_SetRngSeed [▶ 1588]

F_VN_SetRoi [▶ 830]

F_VN_SetRoi_TcVnRectangle_DINT [▶ 831]

F_VN_SetRoi_TcVnRectangle_UDINT [▶ 832]

F_VN_Sigmoid [▶ 1607]

F_VN_SigmoidExp [▶ 1608]

F_VN_SobelFilter [▶ 1350]

F_VN_SobelFilterExp [▶ 1352]

F_VN_SoftMax [▶ 1609]

F_VN_SplitImageChannels [▶ 834]

F_VN_StartAbsWatchdog [▶ 1020]

F_VN_StartAbsWatchdogExp [▶ 1021]

F_VN_StartRelWatchdog [▶ 1022]

F_VN_StartRelWatchdogExp [▶ 1023]

F_VN_StopWatchdog [▶ 1024]

F_VN_SubtractContainers [▶ 522]

F_VN_SubtractImageFromScalar [▶ 560]

F_VN_SubtractImageFromVector [▶ 561]

F_VN_SubtractImages [▶ 562]

F_VN_SubtractScalarFromImage [▶ 563]

F_VN_SubtractVectorFromImage [▶ 564]

F_VN_Threshold [▶ 1363]

F_VN_TrainImageColor [▶ 1290]

F_VN_TrainImageColor_ITcVnMlModel [▶ 1291]

F_VN_TrainImageColorExp [▶ 1293]

F_VN_TrainImageColorExp_ITcVnMlModel [▶ 1295]

F_VN_TrainImageColorExp2 [▶ 1298]

F_VN_TrainImageColorExp2_ITcVnMlModel [▶ 1301]

F_VN_TransformCoordinatesPlanar_Container [▶ 1138]

F_VN_TransformCoordinatesPlanar_Point [▶ 1139]

F_VN_TransformIntoDisplayableImage [▶ 835]

F_VN_TransformIntoDisplayableImageExp [▶ 836]

F_VN_UpdateTimestamp [▶ 1589]

F_VN_UprightBoundingRectangle [▶ 1018]

F_VN_UprightInnerRectangle [▶ 1262]

F_VN_UprightInnerRectangleExp [▶ 1263]

F_VN_VarianceFilter [▶ 1354]

F_VN_VarianceFilterExp [▶ 1355]

F_VN_WarpAffine [▶ 1194]

F_VN_WarpAffine_Container [▶ 1195]

F_VN_WarpAffine_Point [▶ 1196]

F_VN_WarpAffine_Rectangle [▶ 1197]

F_VN_WarpAffineExp [▶ 1198]

F_VN_WarpPerspective [▶ 1199]

F_VN_WarpPerspective_Container [▶ 1200]

F_VN_WarpPerspective_Point [▶ 1201]

F_VN_WarpPerspective_Rectangle [▶ 1202]

F_VN_WarpPerspectiveExp [▶ 1203]

F_VN_WatershedSegmentationExp [▶ 1365]

F_VN_WhiteBalance [▶ 1304]

TC3 Vision Code Reading

F_VN_ReadBarcode [▶ 865]

F_VN_ReadBarcodeExp [▶ 867]

F_VN_ReadBarcodeExp2 [▶ 869]

F_VN_ReadDataMatrixCode [▶ 871]

F_VN_ReadDataMatrixCodeExp [▶ 874]

F_VN_ReadDataMatrixCodeExp2 [▶ 875]

F_VN_ReadDataMatrixCodeRoi [▶ 877]

F_VN_ReadDataMatrixCodeRoiExp [▶ 878]

F_VN_ReadPharmaCode [▶ 880]

F_VN_ReadPharmaCodeExp [▶ 882]

F_VN_ReadPharmaCodeExp2 [▶ 884]

F_VN_ReadQRCode [▶ 885]

F_VN_ReadQRCodeExp [▶ 888]

F_VN_ReadQRCodeExp2 [▶ 889]

TC3 Vision Code Quality

F_VN_GradeBarcode [▶ 845]

F_VN_GradeBarcodeExp [▶ 848]

F_VN_GradeDataMatrixCode [▶ 852]

F_VN_GradeDataMatrixCodeExp [▶ 854]

F_VN_GradeQRCode [▶ 857]

F_VN_GradeQRCodeExp [▶ 860]

TC3 Vision Metrology 2D

F_VN_AdjustSearchWindowOrientationToLinearEdge [▶ 1517]

F_VN_CalibrateCamera [▶ 1102]

F_VN_CalibrateCameraExp [▶ 1105]

F_VN_CalibrateCameraExp2 [▶ 1108]

F_VN_CalibrateCameraExp3 [▶ 1110]

F_VN_CalibrateCameraManually [▶ 1112]

F_VN_CalibrateCameraManuallyExp [▶ 1114]

F_VN_CalibrateLinescanCamera [▶ 1117]

F_VN_CalibrateLinescanCameraExp [▶ 1119]

F_VN_ClosestPointsBF [▶ 1518]

F_VN_CompensateLensDistortion [▶ 1121]

F_VN_CompensateLensDistortionExp1 [▶ 1122]

F_VN_CompensateLensDistortionExp2 [▶ 1123]

F_VN_CompensateLensDistortionForPoints [▶ 1124]

F_VN_CompensateLensDistortionForPointsExp1 [▶ 1125]

F_VN_CompensateLensDistortionForPointsExp2 [▶ 1126]

F_VN_DetectPatternPoints [▶ 1127]

F_VN_DetectPatternPointsExp [▶ 1128]

F_VN_GenerateCalibrationPatternReferencePoints [▶ 1130]

F_VN_ImagePointsWorldDistance [▶ 1131]

F_VN_LocateCircularArc [▶ 1520]

F_VN_LocateCircularArcExp [▶ 1523]

F_VN_LocateEdge [▶ 1526]

F_VN_LocateEdgeExp [▶ 1529]

F_VN_LocateEdges [▶ 1532]

F_VN_LocateEdgesExp [▶ 1535]

F_VN_LocateEllipse [▶ 1538]

F_VN_LocateEllipseExp [▶ 1541]

F_VN_MeasureAngleBetweenEdges [▶ 1544]

F_VN_MeasureAngleBetweenEdgesExp [▶ 1546]

F_VN_MeasureEdgeDistance [▶ 1549]

F_VN_MeasureEdgeDistanceExp [▶ 1551]

F_VN_MeasureMinEdgeDistance [▶ 1554]

F_VN_MeasureMinEdgeDistanceExp [▶ 1556]

F_VN_SolvePnP [▶ 1132]

F_VN_SolvePnPExp [▶ 1134]

F_VN_SortDetectedPatternPoints [▶ 1135]

F_VN_TransformCoordinatesImageToWorld_Container [▶ 1136]

F_VN_TransformCoordinatesImageToWorld_Point [▶ 1137]

F_VN_TransformCoordinatesWorldToImage_Container [▶ 1140]

F_VN_TransformCoordinatesWorldToImage_Point [▶ 1141]

TC3 Vision Matching

F_VN_DrawKeypoints [▶ 1045]

F_VN_DrawKeypointsExp [▶ 1046]

F_VN_DrawMatches [▶ 1057]

F_VN_DrawMatchesExp [▶ 1058]

F_VN_FilterGoodMatches [▶ 1368]

F_VN_FindReferenceKeyPointsInImage [▶ 1369]

F_VN_FindReferenceKeyPointsInImageAKAZE [▶ 1371]

F_VN_FindReferenceKeyPointsInImageAKAZEExp [▶ 1373]

F_VN_FindReferenceKeyPointsInImageBRISK [▶ 1375]

F_VN_FindReferenceKeyPointsInImageBRISKExp [▶ 1377]

F_VN_FindReferenceKeyPointsInImageExp [▶ 1379]

F_VN_FindReferenceKeyPointsInImageORB [▶ 1381]

F_VN_FindReferenceKeyPointsInImageORBExp [▶ 1383]

F_VN_GetMatchCoordinates [▶ 1385]

F_VN_KeyPointsAGAST [▶ 1386]

F_VN_KeyPointsAGASTExp [▶ 1387]

F_VN_KeyPointsAndDescriptorsAKAZE [▶ 1388]

F_VN_KeyPointsAndDescriptorsAKAZEExp [▶ 1388]

F_VN_KeyPointsAndDescriptorsBRISK [▶ 1389]

F_VN_KeyPointsAndDescriptorsBRISKExp [▶ 1390]

F_VN_KeyPointsAndDescriptorsKAZE [▶ 1391]

F_VN_KeyPointsAndDescriptorsKAZEExp [▶ 1392]

F_VN_KeyPointsAndDescriptorsORB [▶ 1393]

F_VN_KeyPointsAndDescriptorsORBExp [▶ 1394]

F_VN_KeyPointsFAST [▶ 1395]

F_VN_KeyPointsFASTExp [▶ 1396]

F_VN_KeyPointsGFTT [▶ 1397]

F_VN_KeyPointsGFTTExp [▶ 1398]

F_VN_KeyPointsMSER [▶ 1399]

F_VN_KeyPointsMSERExp [▶ 1400]

F_VN_KeyPointsSB [▶ 1401]

F_VN_KeyPointsSBExp [▶ 1402]

F_VN_MatchContours [▶ 1013]

F_VN_MatchContours1vsN [▶ 1014]

F_VN_MatchContours1vsNExp [▶ 1016]

F_VN_MatchContoursExp [▶ 1017]

F_VN_MatchDescriptorsBF [▶ 1403]

F_VN_MatchDescriptorsBFExp [▶ 1404]

F_VN_MatchDescriptorsFlannLsh [▶ 1405]

F_VN_MatchDescriptorsFlannLshExp [▶ 1406]

F_VN_MatchDescriptorsKnnBF [▶ 1407]

F_VN_MatchDescriptorsKnnBFExp [▶ 1408]

F_VN_MatchDescriptorsKnnFlannLsh [▶ 1409]

F_VN_MatchDescriptorsKnnFlannLshExp [▶ 1410]

F_VN_MatchImageHuMoments [▶ 1227]

F_VN_MatchTemplate [▶ 1228]

F_VN_MatchTemplateAndEvaluate [▶ 1229]

F_VN_MatchTemplateAndEvaluateExp [▶ 1230]

F_VN_MatchTemplateExp [▶ 1232]

F_VN_RegionsMSER [▶ 1411]

F_VN_RegionsMSERExp [▶ 1412]

TC3 Vision OCR

F_VN_OCR [▶ 1612]

F_VN_OCRExp [▶ 1614]

TC3 Machine Learning Realtime Inference

F_VN_CreateBoostClassifier [▶ 1414]

F_VN_CreateBoostClassifierExp [▶ 1416]

F_VN_CreateBoostClassifierExp2 [▶ 1417]

F_VN_CreateKmppModel [▶ 1419]

F_VN_CreateKmppModelExp [▶ 1421]

F_VN_CreateKnnModel [▶ 1423]

F_VN_CreateLbgModel [▶ 1424]

F_VN_CreateLbgModelExp [▶ 1426]

F_VN_CreateLdaTransform [▶ 1428]

F_VN_CreateLdaTransformViaComponentNum [▶ 1430]

F_VN_CreateNbcModel [▶ 1432]

F_VN_CreatePcaTransform [▶ 1433]

F_VN_CreatePcaTransformViaComponentNum [▶ 1435]

F_VN_CreatePcaTransformViaVariance [▶ 1436]

F_VN_CreateRTreesModel [▶ 1438]

F_VN_CreateRTreesModelExp [▶ 1439]

F_VN_CreateRTreesModelExp2 [▶ 1441]

F_VN_CreateStaModel [▶ 1443]

F_VN_CreateStaModelExp [▶ 1445]

F_VN_CreateStaModelExp2 [▶ 1447]

F_VN_CreateSvmModel [▶ 1449]

F_VN_CreateSvmModelExp [▶ 1452]

F_VN_CreateSvmModelExp2 [▶ 1455]

F_VN_CreateSvmSgdClassifier [▶ 1458]

F_VN_CreateSvmSgdClassifierExp [▶ 1459]

F_VN_FeatureScaling [▶ 1461]

F_VN_FeatureScalingExp [▶ 1463]

F_VN_FeatureTransform [▶ 1465]

F_VN_GetBatchClusters [▶ 1466]

F_VN_GetBatchClustersExp [▶ 1467]

F_VN_GetBatchNovelties [▶ 1469]

F_VN_GetClusterCenter [▶ 1470]

F_VN_GetClusterNum [▶ 1472]

F_VN_GetFeatureScales [▶ 1473]

F_VN_GetSampleCluster [▶ 1475]

F_VN_GetSampleClusterExp [▶ 1476]

F_VN_GetSampleNovelty [▶ 1478]

F_VN_Granulometry [▶ 1617]

F_VN_HaralickFeatures [▶ 1619]

F_VN_InverseFeatureScaling [▶ 1479]

F_VN_InverseFeatureScaling_REAL [▶ 1481]

F_VN_InverseFeatureScalingExp [▶ 1483]

F_VN_InverseFeatureScalingExp_REAL [▶ 1484]

F_VN_InverseFeatureTransform [▶ 1486]

F_VN_PredictBatch [▶ 1487]

F_VN_PredictBatchExp [▶ 1489]

F_VN_PredictSampleClass [▶ 1491]

F_VN_PredictSampleClassExp [▶ 1493]

F_VN_PredictSampleScalar [▶ 1494]

F_VN_PredictSampleScalarExp [▶ 1496]

F_VN_PredictSampleVector [▶ 1497]

F_VN_PredictSampleVectorExp [▶ 1499]

F_VN_TrainBatch [▶ 1500]

F_VN_TrainBatchClusters [▶ 1502]

F_VN_TrainSample [▶ 1504]

F_VN_TrainSampleClass [▶ 1505]

F_VN_TrainSampleCluster [▶ 1507]

F_VN_TrainSampleScalar [▶ 1508]

F_VN_TrainSampleVector [▶ 1510]

TC3 Neural Network Realtime Inference

F_VN_ExecuteNeuralNetwork [▶ 1592]

F_VN_ExecuteNeuralNetwork_MI [▶ 1593]

F_VN_ExecuteNeuralNetwork_MIMO_String [▶ 1594]

F_VN_ExecuteNeuralNetwork_MISO [▶ 1595]

F_VN_ExecuteNeuralNetwork_MO [▶ 1596]

F_VN_ExecuteNeuralNetwork_MultiLayerOutput [▶ 1597]

F_VN_ExecuteNeuralNetwork_SingleLayerOutput [▶ 1598]

F_VN_ExecuteNeuralNetwork_SO [▶ 1599]

F_VN_GetNeuralNetworkLayerNames [▶ 1600]

F_VN_GetNeuralNetworkLayerNamesExp [▶ 1601]

F_VN_GetNeuralNetworkLayerNamesExp_String [▶ 1601]

6.1.4.2 Algebraic Container Operations

Diese Gruppe enthält Funktionen zur elementweisen Anwendung von algebraischen Operatoren auf Containern [▶ 163].

Funktionen

Algebra

- F_VN_AddContainers [▶ 509]
- F_VN_AddToContainerElements [▶ 481]
- F_VN_DivideContainers(Exp) [▶ 513]
- F_VN_MultiplyContainers [▶ 517]
- F_VN_MultiplyWithContainerElements [▶ 518]
- F_VN_NegateContainer [▶ 521]
- F_VN_SubtractContainers [▶ 522]

Bitweise Operatoren

- F_VN_BitwiseAndContainers [▶ 510]
- F_VN_BitwiseNotContainer [▶ 511]
- F_VN_BitwiseOrContainers [▶ 511]

- F_VN_BitwiseXorContainers [▶ 512]

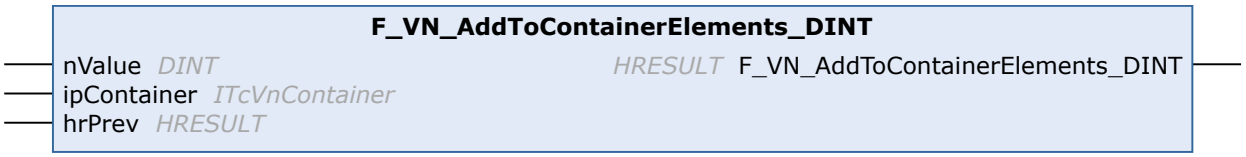
Statistische Merkmale

- F_VN_MaxContainer [▶ 515]
- F_VN_MinContainer [▶ 516]

6.1.4.2.1 F_VN_AddToContainerElements

Dieses Kapitel enthält Funktionen, mit denen ein Wert zu jedem einzelnen Container-Element entsprechend dem Datentyp addiert werden kann.

6.1.4.2.1.1 F_VN_AddToContainerElements_DINT



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_DINT : HRESULT
VAR_INPUT
    nValue      : DINT;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nValue	DINT	Value
ipContainer	ITcVnContainer [▶ 385]	Container with DINT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

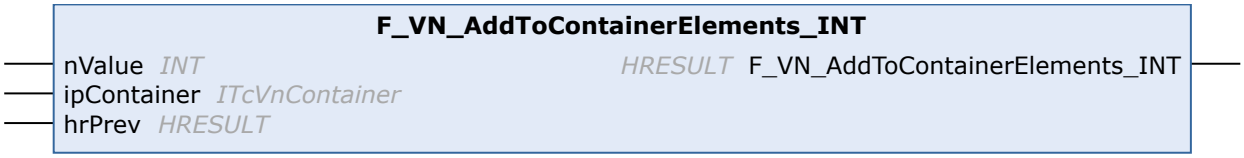
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.2 F_VN_AddToContainerElements_INT



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_INT : HRESULT
VAR_INPUT
    nValue      : INT;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nValue	INT	Value
ipContainer	ITcVnContainer [▶ 385]	Container with INT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

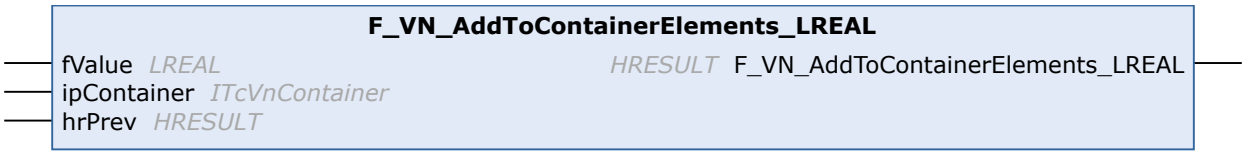
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.3 F_VN_AddToContainerElements_LREAL



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_LREAL : HRESULT
VAR_INPUT
    fValue      : LREAL;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fValue	LREAL	Value
ipContainer	ITcVnContainer [▶ 385]	Container with LREAL elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

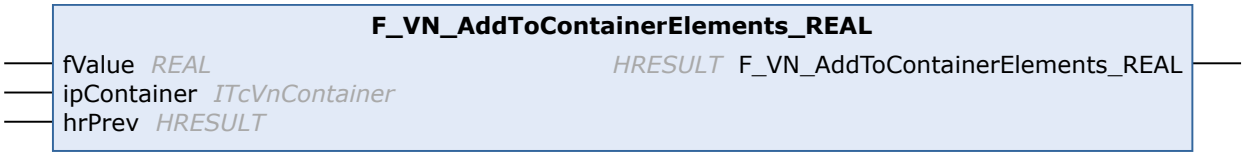
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.4 F_VN_AddToContainerElements_REAL



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_REAL : HRESULT
VAR_INPUT
    fValue      : REAL;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fValue	REAL	Value
ipContainer	ITcVnContainer [▶ 385]	Container with REAL elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.5 F_VN_AddToContainerElements_SINT

Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_SINT : HRESULT
VAR_INPUT
    nValue      : SINT;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nValue	SINT	Value
ipContainer	ITcVnContainer [▶ 385]	Container with SINT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

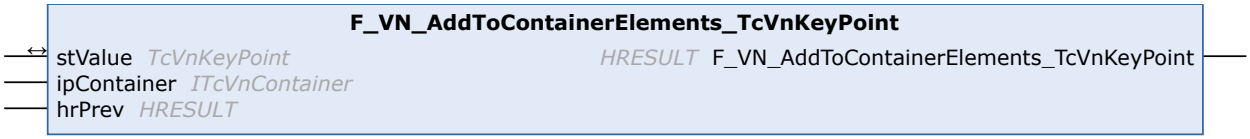
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.6 F_VN_AddToContainerElements_TcVnKeyPoint



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnKeyPoint : HRESULT
VAR_IN_OUT
    stValue      : TcVnKeyPoint;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnKeyPoint elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stValue	TcVnKeyPoint [▶ 244]	Value

Return value

HRESULT [▶ 150]

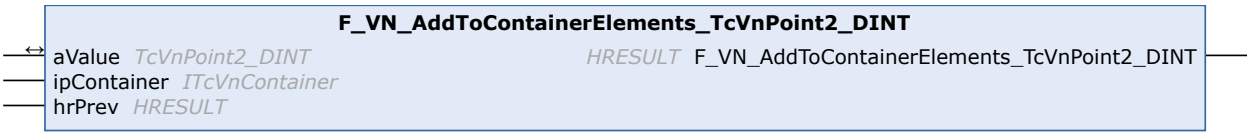
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.7 F_VN_AddToContainerElements_TcVnPoint2_DINT



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnPoint2_DINT : HRESULT
VAR_IN_OUT
    aValue      : TcVnPoint2_DINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnPoint2_DINT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint2_DINT [▶ 171]	Value

Return value

HRESULT [▶ 150]

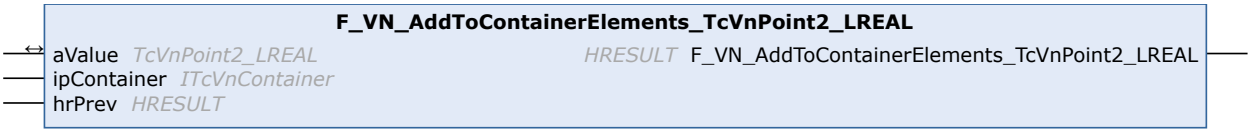
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.8 F_VN_AddToContainerElements_TcVnPoint2_LREAL



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnPoint2_LREAL : HRESULT
VAR_IN_OUT
    aValue      : TcVnPoint2_LREAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnPoint2_LREAL elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint2_LREAL [▶ 171]	Value

Return value

HRESULT [▶ 150]

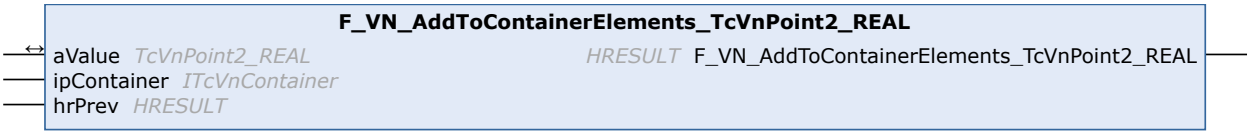
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.9 F_VN_AddToContainerElements_TcVnPoint2_REAL



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnPoint2_REAL : HRESULT
VAR_IN_OUT
    aValue      : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnPoint2_REAL elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint2_REAL [▶ 171]	Value

Return value

HRESULT [▶ 150]

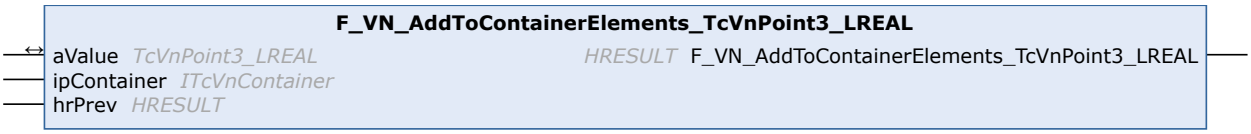
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.10 F_VN_AddToContainerElements_TcVnPoint3_LREAL



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnPoint3_LREAL : HRESULT
VAR_IN_OUT
    aValue      : TcVnPoint3_LREAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnPoint3_LREAL elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint3_LREAL [▶ 171]	Value

Return value

HRESULT [▶ 150]

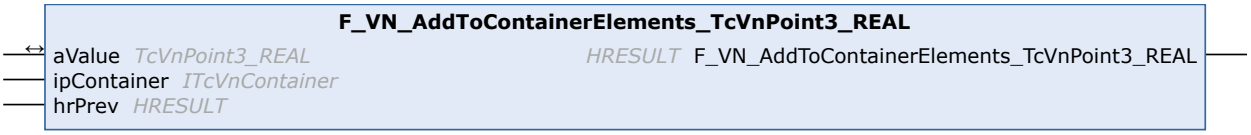
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.11 F_VN_AddToContainerElements_TcVnPoint3_REAL



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnPoint3_REAL : HRESULT
VAR_IN_OUT
    aValue      : TcVnPoint3_REAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnPoint3_REAL elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint3_REAL [▶ 171]	Value

Return value

HRESULT [▶ 150]

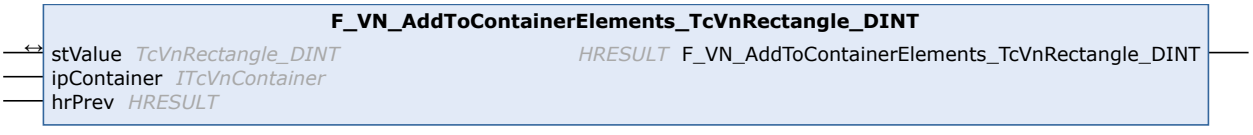
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.12 F_VN_AddToContainerElements_TcVnRectangle_DINT



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnRectangle_DINT : HRESULT
VAR_IN_OUT
    stValue     : TcVnRectangle_DINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnRectangle_DINT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stValue	TcVnRectangle_DINT [▶ 257]	Value

Return value

HRESULT [▶ 150]

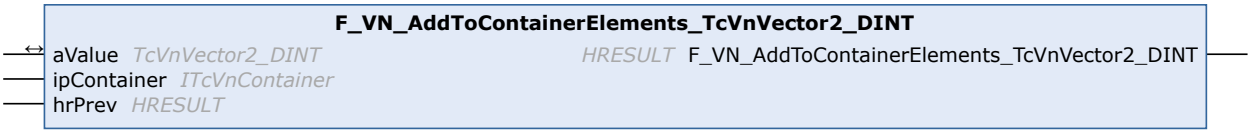
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.13 F_VN_AddToContainerElements_TcVnVector2_DINT



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnVector2_DINT : HRESULT
VAR_IN_OUT
    aValue      : TcVnVector2_DINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector2_DINT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_DINT [▶ 173]	Value

Return value

HRESULT [▶ 150]

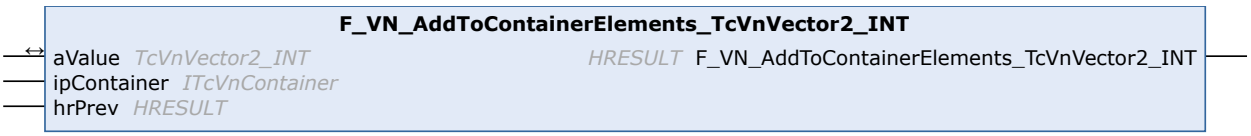
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.14 F_VN_AddToContainerElements_TcVnVector2_INT



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnVector2_INT : HRESULT
VAR_IN_OUT
    aValue      : TcVnVector2_INT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector2_INT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_INT [▶ 173]	Value

Return value

HRESULT [▶ 150]

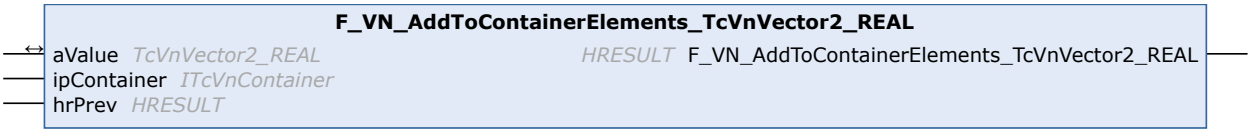
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.15 F_VN_AddToContainerElements_TcVnVector2_REAL



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnVector2_REAL : HRESULT
VAR_IN_OUT
    aValue      : TcVnVector2_REAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector2_REAL elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_REAL [▶ 173]	Value

Return value

HRESULT [▶ 150]

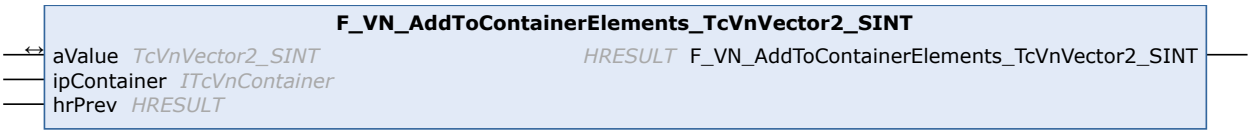
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.16 F_VN_AddToContainerElements_TcVnVector2_SINT



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnVector2_SINT : HRESULT
VAR_IN_OUT
    aValue      : TcVnVector2_SINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector2_SINT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_SINT [▶ 173]	Value

Return value

HRESULT [▶ 150]

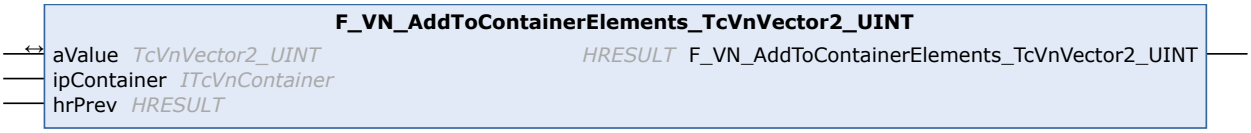
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.17 F_VN_AddToContainerElements_TcVnVector2_UINT



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnVector2_UINT : HRESULT
VAR_IN_OUT
    aValue      : TcVnVector2_UINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector2_UINT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_UINT [▶ 173]	Value

Return value

HRESULT [▶ 150]

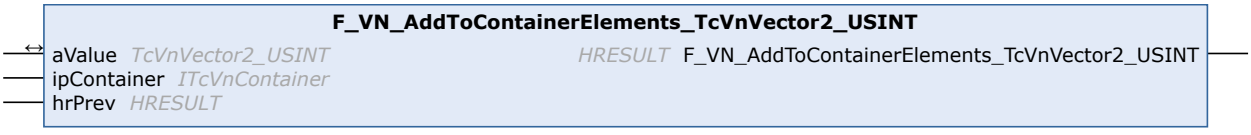
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.18 F_VN_AddToContainerElements_TcVnVector2_USINT



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnVector2_USINT : HRESULT
VAR_IN_OUT
    aValue      : TcVnVector2_USINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector2_USINT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_USINT [▶ 173]	Value

Return value

HRESULT [▶ 150]

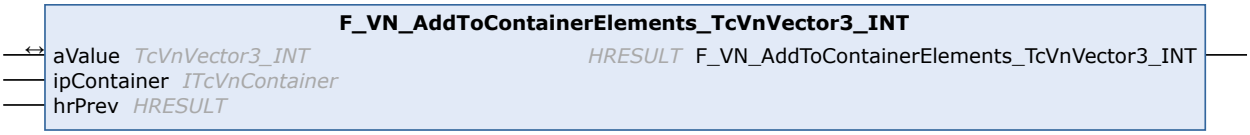
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.19 F_VN_AddToContainerElements_TcVnVector3_INT



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnVector3_INT : HRESULT
VAR_IN_OUT
    aValue      : TcVnVector3_INT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector3_INT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_INT [▶ 173]	Value

Return value

HRESULT [▶ 150]

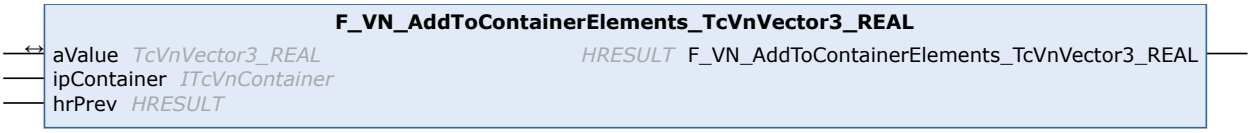
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.20 F_VN_AddToContainerElements_TcVnVector3_REAL



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnVector3_REAL : HRESULT
VAR_IN_OUT
    aValue      : TcVnVector3_REAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector3_REAL elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_REAL [▶ 173]	Value

Return value

HRESULT [▶ 150]

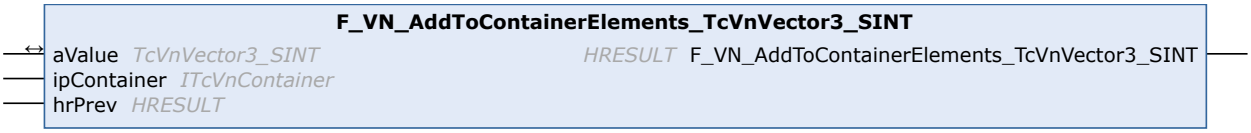
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.21 F_VN_AddToContainerElements_TcVnVector3_SINT



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnVector3_SINT : HRESULT
VAR_IN_OUT
    aValue      : TcVnVector3_SINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector3_SINT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_SINT [▶ 173]	Value

Return value

HRESULT [▶ 150]

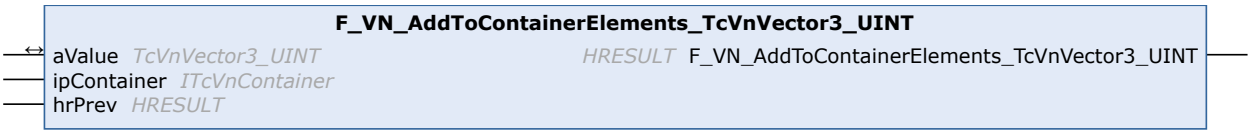
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.22 F_VN_AddToContainerElements_TcVnVector3_UINT



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnVector3_UINT : HRESULT
VAR_IN_OUT
    aValue      : TcVnVector3_UINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector3_UINT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_UINT [▶ 173]	Value

Return value

HRESULT [▶ 150]

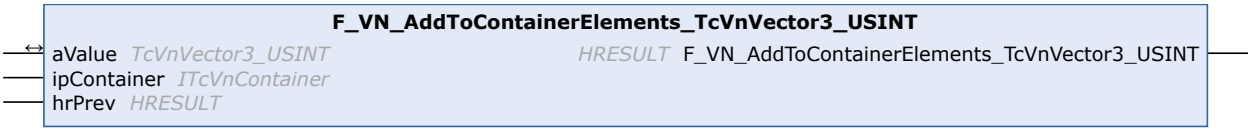
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.23 F_VN_AddToContainerElements_TcVnVector3_USINT



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnVector3_USINT : HRESULT
VAR_IN_OUT
    aValue      : TcVnVector3_USINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector3_USINT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_USINT [▶ 173]	Value

Return value

HRESULT [▶ 150]

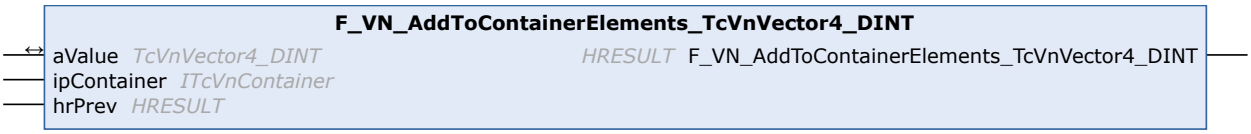
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.24 F_VN_AddToContainerElements_TcVnVector4_DINT



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnVector4_DINT : HRESULT
VAR_IN_OUT
    aValue      : TcVnVector4_DINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector4_DINT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_DINT [▶ 173]	Value

Return value

HRESULT [▶ 150]

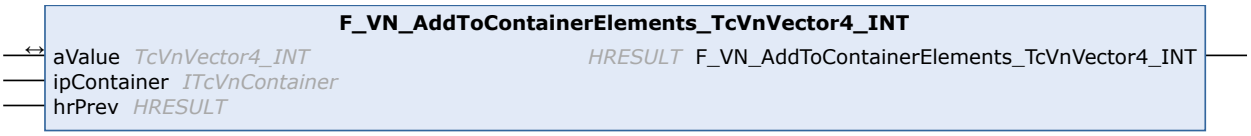
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.25 F_VN_AddToContainerElements_TcVnVector4_INT



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnVector4_INT : HRESULT
VAR_IN_OUT
    aValue      : TcVnVector4_INT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector4_INT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_INT [▶ 173]	Value

Return value

HRESULT [▶ 150]

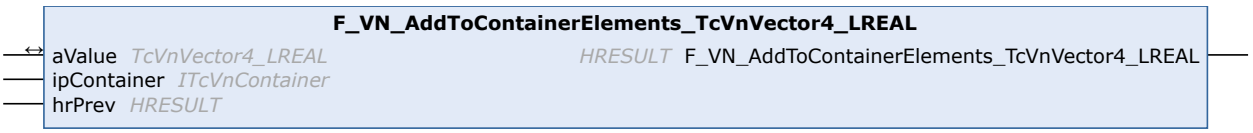
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.26 F_VN_AddToContainerElements_TcVnVector4_LREAL



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnVector4_LREAL : HRESULT
VAR_IN_OUT
    aValue      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector4_LREAL elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_LREAL [▶ 173]	Value

Return value

HRESULT [▶ 150]

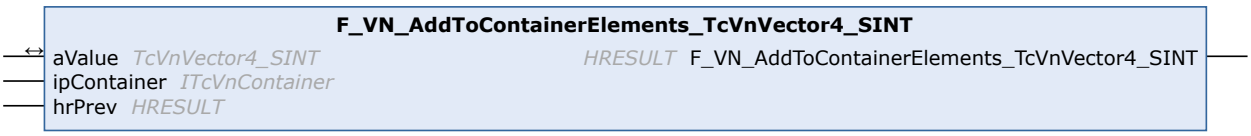
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.27 F_VN_AddToContainerElements_TcVnVector4_SINT



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnVector4_SINT : HRESULT
VAR_IN_OUT
    aValue      : TcVnVector4_SINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector4_SINT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_SINT [▶ 173]	Value

Return value

HRESULT [▶ 150]

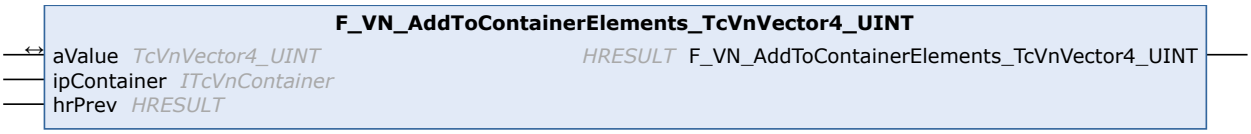
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.28 F_VN_AddToContainerElements_TcVnVector4_UINT



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnVector4_UINT : HRESULT
VAR_IN_OUT
    aValue      : TcVnVector4_UINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector4_UINT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_UINT [▶ 173]	Value

Return value

HRESULT [▶ 150]

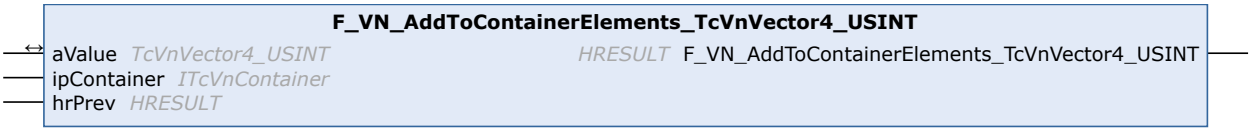
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.29 F_VN_AddToContainerElements_TcVnVector4_USINT



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnVector4_USINT : HRESULT
VAR_IN_OUT
    aValue      : TcVnVector4_USINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector4_USINT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_USINT [▶ 173]	Value

Return value

HRESULT [▶ 150]

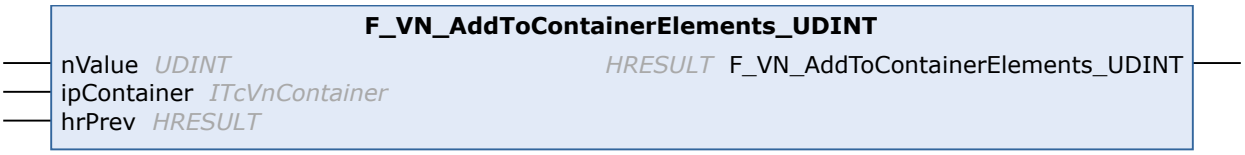
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.30 F_VN_AddToContainerElements_UDINT



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_UDINT : HRESULT
VAR_INPUT
    nValue      : UDINT;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nValue	UDINT	Value
ipContainer	ITcVnContainer [▶ 385]	Container with UDINT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

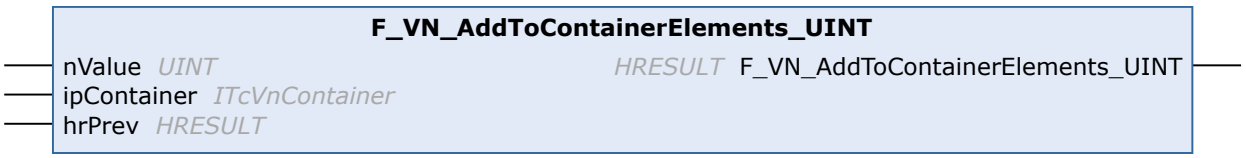
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.31 F_VN_AddToContainerElements_UINT



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_UINT : HRESULT
VAR_INPUT
    nValue      : UINT;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nValue	UINT	Value
ipContainer	ITcVnContainer [▶ 385]	Container with UINT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

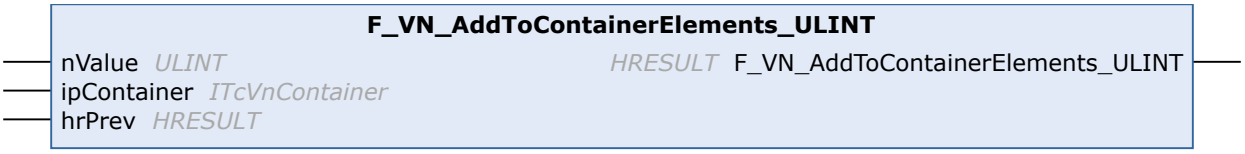
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.32 F_VN_AddToContainerElements_ULINT



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_ULINT : HRESULT
VAR_INPUT
    nValue      : ULINT;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nValue	ULINT	Value
ipContainer	ITcVnContainer [▶ 385]	Container with ULINT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

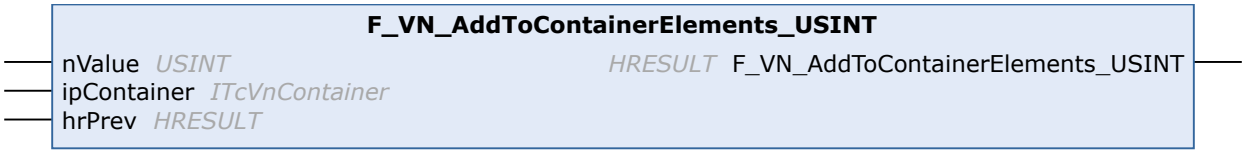
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.33 F_VN_AddToContainerElements_USINT



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_USINT : HRESULT
VAR_INPUT
    nValue      : USINT;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nValue	USINT	Value
ipContainer	ITcVnContainer [▶ 385]	Container with USINT elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

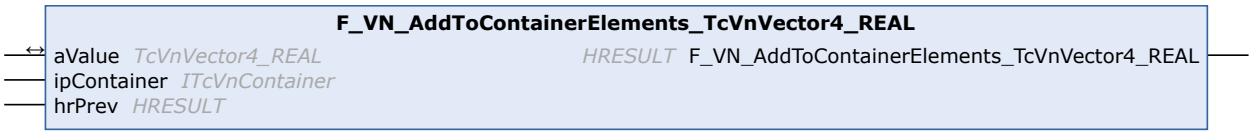
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.34 F_VN_AddToContainerElements_TcVnVector4_REAL



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnVector4_REAL : HRESULT
VAR_IN_OUT
    aValue      : TcVnVector4_REAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector4_REAL elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_REAL [▶ 173]	Value

Return value

HRESULT [▶ 150]

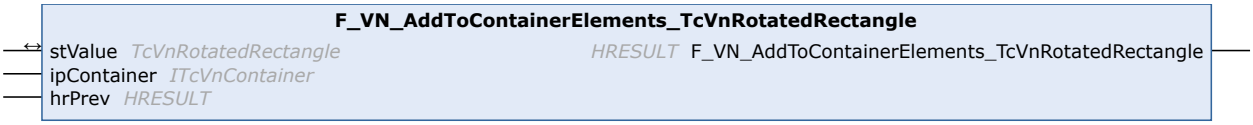
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.1.35 F_VN_AddToContainerElements_TcVnRotatedRectangle



Add a value to each container element.

Syntax

Definition:

```
FUNCTION F_VN_AddToContainerElements_TcVnRotatedRectangle : HRESULT
VAR_IN_OUT
    stValue     : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnRotatedRectangle elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

/ In/Outputs

Name	Type	Description
stValue	TcVnRotatedRectangle [▶ 258]	Value

Return value

HRESULT [▶ 150]

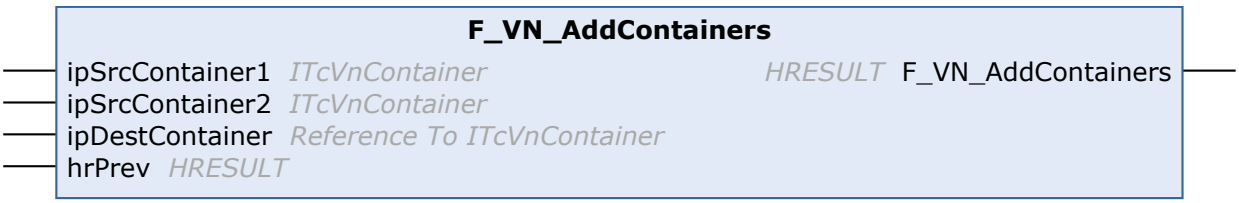
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.2 F_VN_AddContainers



Element-wise addition of two containers (same length and type).

Syntax

Definition:

```
FUNCTION F_VN_AddContainers : HRESULT
VAR_INPUT
    ipSrcContainer1 : ITcVnContainer;
    ipSrcContainer2 : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer1	ITcVnContainer [▶ 385]	Source container 1
ipSrcContainer2	ITcVnContainer [▶ 385]	Source container 2
ipDestContainer	Reference To ITcVnContainer [▶ 385]	Returns the resulting container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

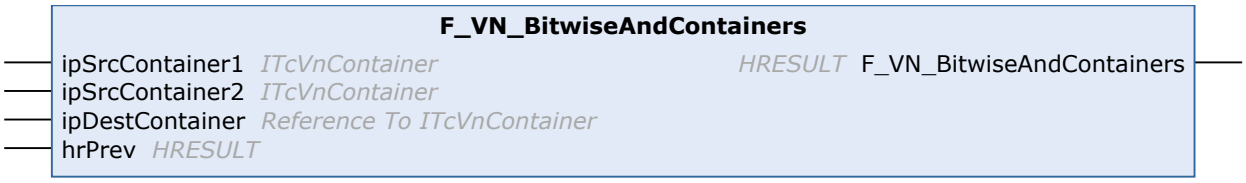
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.3 F_VN_BitwiseAndContainers



Element-wise application of a bit-wise AND operator to two containers (same length and type, integer only).

Syntax

Definition:

```
FUNCTION F_VN_BitwiseAndContainers : HRESULT
VAR_INPUT
    ipSrcContainer1 : ITcVnContainer;
    ipSrcContainer2 : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer1	ITcVnContainer [▶ 385]	Source container 1
ipSrcContainer2	ITcVnContainer [▶ 385]	Source container 2
ipDestContainer	Reference To ITcVnContainer [▶ 385]	Returns the resulting container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

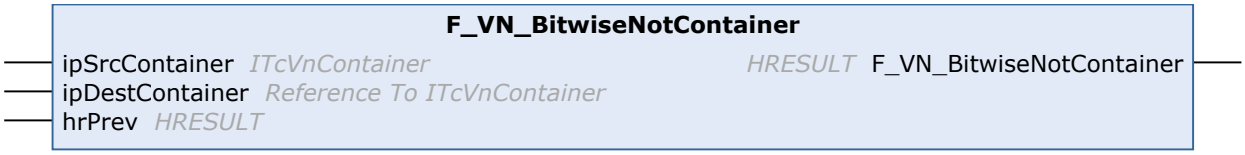
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.4 F_VN_BitwiseNotContainer



Element-wise application of a bit-wise NOT operator to a container (integer only).

Syntax

Definition:

```
FUNCTION F_VN_BitwiseNotContainer : HRESULT
VAR_INPUT
    ipSrcContainer  : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer	ITcVnContainer [▶ 385]	Source container
ipDestContainer	Reference To ITcVnContainer [▶ 385]	Returns the resulting container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.5 F_VN_BitwiseOrContainers

Element-wise application of a bit-wise OR operator to two containers (same length and type, integer only).

Syntax

Definition:

```
FUNCTION F_VN_BitwiseOrContainers : HRESULT
VAR_INPUT
    ipSrcContainer1 : ITcVnContainer;
    ipSrcContainer2 : ITcVnContainer;
```

```
    ipDestContainer : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer 1	ITcVnContainer [▶ 385]	Source container 1
ipSrcContainer 2	ITcVnContainer [▶ 385]	Source container 2
ipDestContaine r	Reference To ITcVnContainer [▶ 385]	Returns the resulting container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

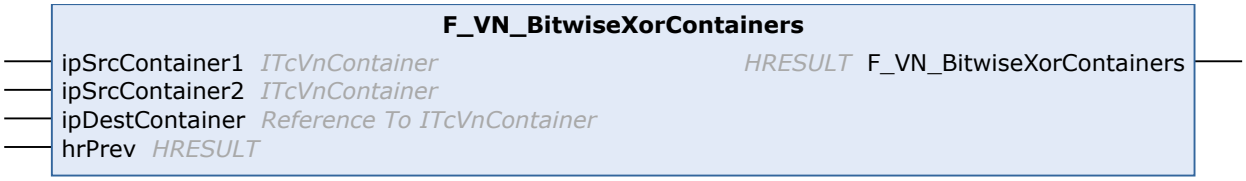
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.6 F_VN_BitwiseXorContainers



Element-wise application of a bit-wise XOR operator to two containers (same length and type, integer only).

Syntax

Definition:

```
FUNCTION F_VN_BitwiseXorContainers : HRESULT
VAR_INPUT
    ipSrcContainer1 : ITcVnContainer;
    ipSrcContainer2 : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer 1	ITcVnContainer [▶ 385]	Source container 1
ipSrcContainer 2	ITcVnContainer [▶ 385]	Source container 2
ipDestContaine r	Reference To ITcVnContainer [▶ 385]	Returns the resulting container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

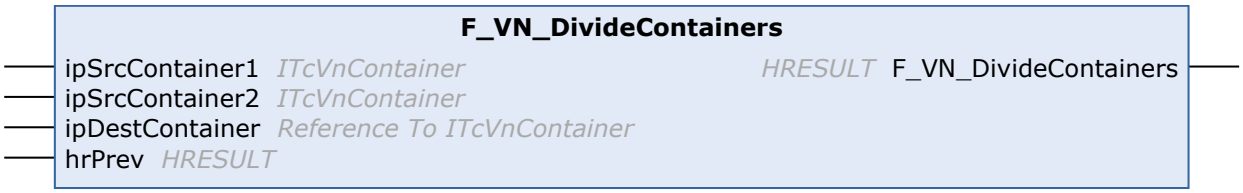
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.7 F_VN_DivideContainers



Element-wise division of two containers (same length and type). In case of integer division by zero, the resulting element is set to 0 and S_DIVISION_BY_ZERO is returned instead of S_OK (value can be changed in F_VN_DivideContainersExp).

Syntax

Definition:

```
FUNCTION F_VN_DivideContainers : HRESULT
VAR_INPUT
    ipSrcContainer1 : ITcVnContainer;
    ipSrcContainer2 : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer 1	ITcVnContainer [▶ 385]	Source container 1
ipSrcContainer 2	ITcVnContainer [▶ 385]	Source container 2
ipDestContaine r	Reference To ITcVnContainer [▶ 385]	Returns the resulting container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

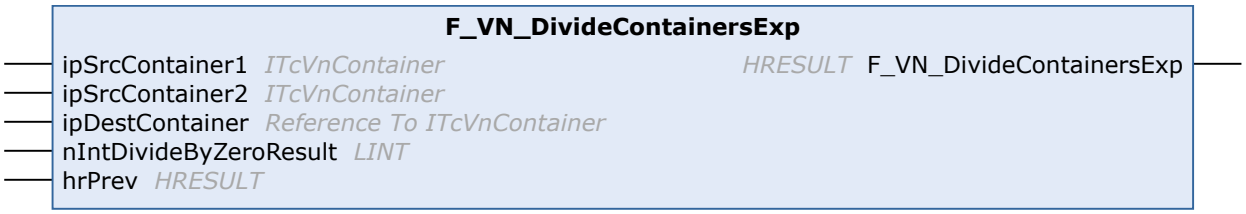
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.8 F_VN_DivideContainersExp



Element-wise division of two containers (same length and type). In case of integer division by zero, the resulting element is set to nIntDivideByZeroResult and S_DIVISION_BY_ZERO is returned instead of S_OK. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_DivideContainersExp : HRESULT
VAR_INPUT
    ipSrcContainer1        : ITcVnContainer;
    ipSrcContainer2        : ITcVnContainer;
    ipDestContainer        : Reference To ITcVnContainer;
    nIntDivideByZeroResult : LINT;
    hrPrev                 : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer1	ITcVnContainer [▶ 385]	Source container 1
ipSrcContainer2	ITcVnContainer [▶ 385]	Source container 2
ipDestContainer	Reference To ITcVnContainer [▶ 385]	Returns the resulting container
nIntDivideByZeroResult	LINT	Value set as result in case of integer division by zero
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

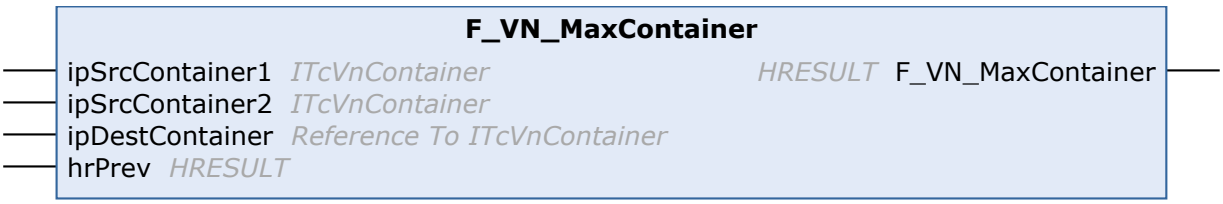
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.9 F_VN_MaxContainer



Element-wise maximum of two containers (same length and type).

Syntax

Definition:

```
FUNCTION F_VN_MaxContainer : HRESULT
VAR_INPUT
    ipSrcContainer1 : ITcVnContainer;
    ipSrcContainer2 : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer1	ITcVnContainer [▶ 385]	Source container 1
ipSrcContainer2	ITcVnContainer [▶ 385]	Source container 2
ipDestContainer	Reference To ITcVnContainer [▶ 385]	Returns the resulting container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.10 F_VN_MinContainer

Element-wise minimum of two containers (same length and type).

Syntax

Definition:

```
FUNCTION F_VN_MinContainer : HRESULT
VAR_INPUT
    ipSrcContainer1 : ITcVnContainer;
    ipSrcContainer2 : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer1	ITcVnContainer [▶ 385]	Source container 1
ipSrcContainer2	ITcVnContainer [▶ 385]	Source container 2
ipDestContainer	Reference To ITcVnContainer [▶ 385]	Returns the resulting container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.11 F_VN_MultiplyContainers

Element-wise multiplication of two containers (same length and type).

Syntax

Definition:

```
FUNCTION F_VN_MultiplyContainers : HRESULT
VAR_INPUT
    ipSrcContainer1 : ITcVnContainer;
    ipSrcContainer2 : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer1	ITcVnContainer [▶ 385]	Source container 1
ipSrcContainer2	ITcVnContainer [▶ 385]	Source container 2
ipDestContainer	Reference To ITcVnContainer [▶ 385]	Returns the resulting container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

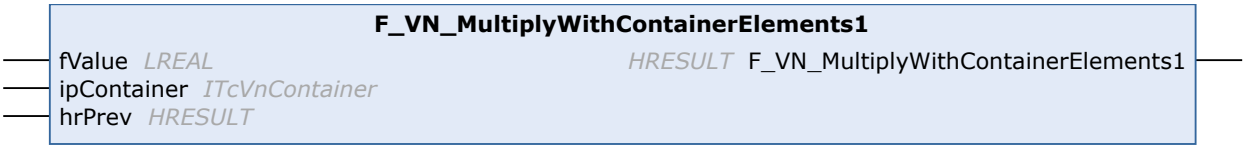
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.12 F_VN_MultiplyWithContainerElements1



Multiply each container element with a value.

Syntax

Definition:

```
FUNCTION F_VN_MultiplyWithContainerElements1 : HRESULT
VAR_INPUT
    fValue      : LREAL;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fValue	LREAL	Value
ipContainer	ITcVnContainer [▶ 385]	Container with 1-dimensional elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

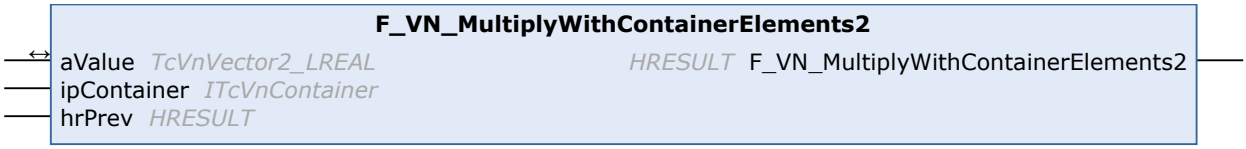
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.13 F_VN_MultiplyWithContainerElements2



Multiply each container element with a value.

Syntax

Definition:

```
FUNCTION F_VN_MultiplyWithContainerElements2 : HRESULT
VAR_IN_OUT
    aValue      : TcVnVector2_LREAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with 2-dimensional elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_LREAL [▶ 173]	Value

Return value

HRESULT [▶ 150]

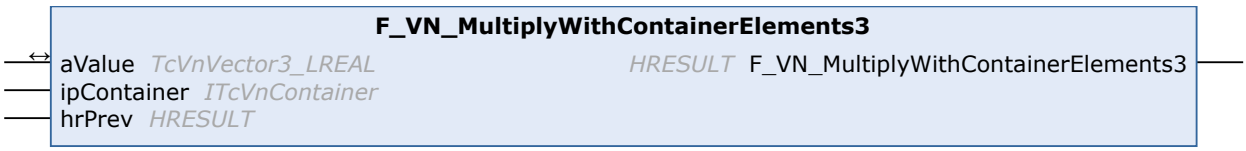
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.14 F_VN_MultiplyWithContainerElements3



Multiply each container element with a value.

Syntax

Definition:

```
FUNCTION F_VN_MultiplyWithContainerElements3 : HRESULT
VAR_IN_OUT
    aValue      : TcVnVector3_LREAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with 3-dimensional elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_LREAL [▶ 173]	Value

Return value

HRESULT [▶ 150]

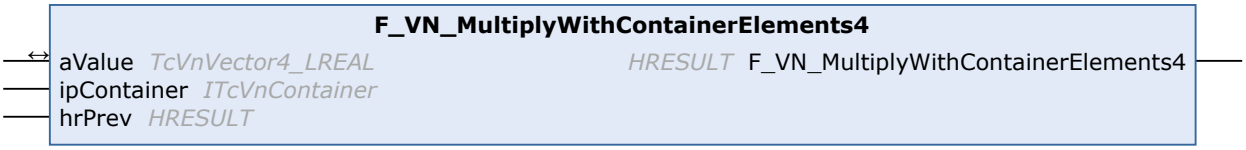
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.15 F_VN_MultiplyWithContainerElements4



Multiply each container element with a value.

Syntax

Definition:

```
FUNCTION F_VN_MultiplyWithContainerElements4 : HRESULT
VAR_IN_OUT
    aValue      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with 4-dimensional elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_LREAL [▶ 173]	Value

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.16 F_VN_NegateContainer

Element-wise negation of a container (two's complement).

Syntax

Definition:

```
FUNCTION F_VN_NegateContainer : HRESULT
VAR_INPUT
    ipSrcContainer  : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer	ITcVnContainer [▶ 385]	Source container
ipDestContainer	Reference To ITcVnContainer [▶ 385]	Returns the resulting container (same type as ipSrcContainer)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

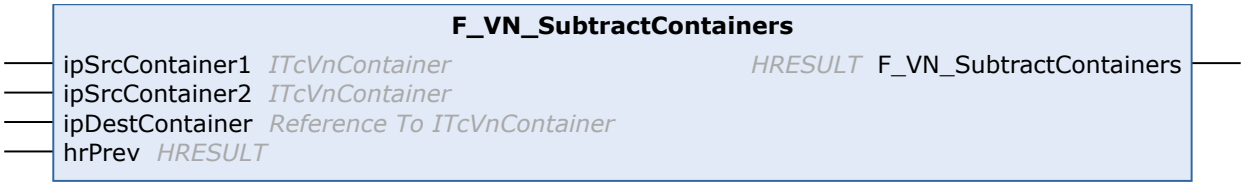
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.2.17 F_VN_SubtractContainers



Element-wise subtraction of two containers (same length and type).

Syntax

Definition:

```
FUNCTION F_VN_SubtractContainers : HRESULT
VAR_INPUT
    ipSrcContainer1 : ITcVnContainer;
    ipSrcContainer2 : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer1	ITcVnContainer [▶ 385]	Source container 1
ipSrcContainer2	ITcVnContainer [▶ 385]	Source container 2
ipDestContainer	Reference To ITcVnContainer [▶ 385]	Returns the resulting container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3 Algebraic Image Operations

Diese Gruppe enthält Funktionen zur pixelweisen Anwendung von algebraischen Operationen auf Bildern [▶ 159].

Funktionen

Addition

- F_VN_AddImages [▶ 524]
- F_VN_AddImagesWeighted(Exp) [▶ 525]
- F_VN_AddScalarToImage [▶ 527]
- F_VN_AddVectorToImage [▶ 528]

Subtraktion

- F_VN_SubtractImages [▶ 562]
- F_VN_SubtractImageFromScalar [▶ 560]
- F_VN_SubtractImageFromVector [▶ 561]
- F_VN_SubtractScalarFromImage [▶ 563]
- F_VN_SubtractVectorFromImage [▶ 564]

Multiplikation

- F_VN_MultiplyImages [▶ 558]
- F_VN_MultiplyImageWithScalar [▶ 559]
- F_VN_MultiplyImageWithVector [▶ 559]

Division

- F_VN_DivideImages [▶ 543]
- F_VN_DivideImageByScalar [▶ 541]
- F_VN_DivideImageByVector [▶ 542]
- F_VN_DivideScalarByImage [▶ 544]
- F_VN_DivideVectorByImage [▶ 545]

Potenzieren und Logarithmieren

- F_VN_ElementwiseExp [▶ 546]
- F_VN_ElementwiseLog [▶ 546]

Bitweise AND

- F_VN_BitwiseAndImages(Exp) [▶ 529]
- F_VN_BitwiseAndScalarWithImage [▶ 530]
- F_VN_BitwiseAndVectorWithImage [▶ 531]

Bitweise NOT

- F_VN_BitwiseNotImage(Exp) [▶ 532]

Bitweise OR

- F_VN_BitwiseOrImages(Exp) [▶ 534]
- F_VN_BitwiseOrScalarWithImage [▶ 535]
- F_VN_BitwiseOrVectorWithImage [▶ 536]

Bitweise XOR

- F_VN_BitwiseXorImages(Exp) [▶ 537]
- F_VN_BitwiseXorScalarWithImage [▶ 539]
- F_VN_BitwiseXorVectorWithImage [▶ 539]

Überblendung

- F_VN_BlendImages [▶ 540]

Maximum

- F_VN_MaxImage(Exp) [▶ 547]
- F_VN_MaxImageWithScalar(Exp) [▶ 549]
- F_VN_MaxImageWithVector(Exp) [▶ 551]

Minimum

- F_VN_MinImage(Exp) [▶ 553]
- F_VN_MinImageWithScalar(Exp) [▶ 554]
- F_VN_MinImageWithVector(Exp) [▶ 556]

6.1.4.3.1 F_VN_AddImages

Element-wise addition of two images using saturation arithmetics.

Syntax

Definition:

```
FUNCTION F_VN_AddImages : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
    ipSrcImage2 : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage1	ITcVnImage [▶ 426]	First source image
ipSrcImage2	ITcVnImage [▶ 426]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Beispiele

- Bildbereiche kopieren [▶ 2843]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.2 F_VN_AddImagesWeighted

Weighted, element-wise addition of two images using saturation arithmetics.

Syntax

Definition:

```
FUNCTION F_VN_AddImagesWeighted : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
    ipSrcImage2 : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    fWeight1    : LREAL;
    fWeight2    : LREAL;
    fDelta      : LREAL;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage1	ITcVnImage [▶ 426]	First source image
ipSrcImage2	ITcVnImage [▶ 426]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
fWeight1	LREAL	Weight factor applied to ipSrcImage1
fWeight2	LREAL	Weight factor applied to ipSrcImage2
fDelta	LREAL	Value added to the weighted sum of both images
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

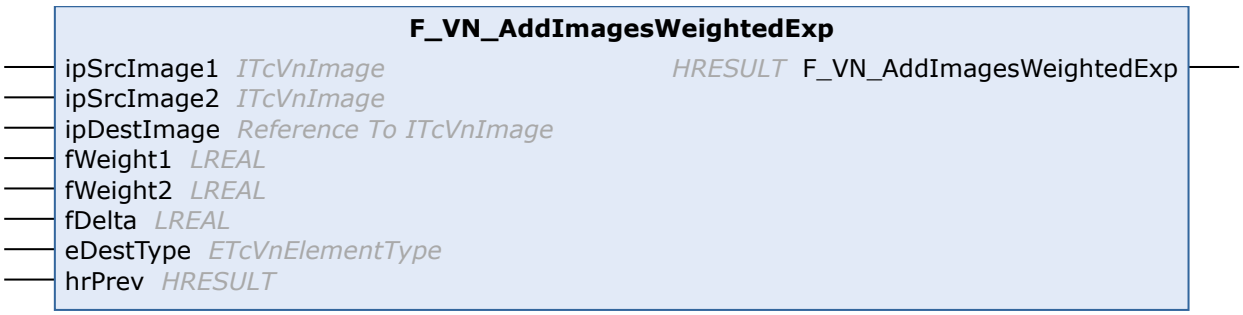
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.3 F_VN_AddImagesWeightedExp



Weighted, element-wise addition of two images using saturation arithmetics.

Syntax

Definition:

```
FUNCTION F_VN_AddImagesWeightedExp : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
    ipSrcImage2 : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    fWeight1    : LREAL;
    fWeight2    : LREAL;
    fDelta      : LREAL;
    eDestType   : ETcVnElementType;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage1	ITcVnImage [▶ 426]	First source image
ipSrcImage2	ITcVnImage [▶ 426]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
fWeight1	LREAL	Weight factor applied to ipSrcImage1
fWeight2	LREAL	Weight factor applied to ipSrcImage2
fDelta	LREAL	Value added to the weighted sum of both images
eDestType	ETcVnElementType [▶ 210]	Destination image depth
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

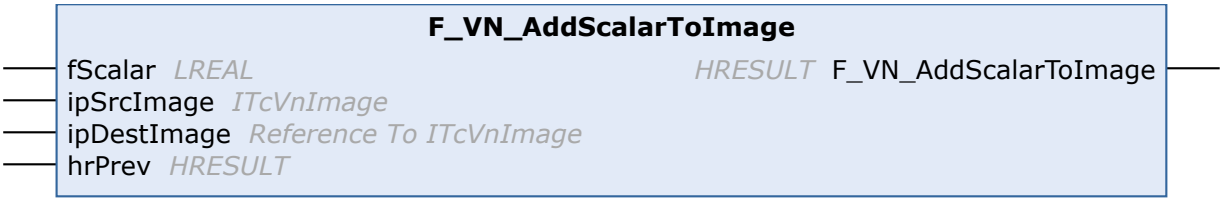
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.4 F_VN_AddScalarToImage



Add a scalar value to each image pixel using saturation arithmetics.

Syntax

Definition:

```
FUNCTION F_VN_AddScalarToImage : HRESULT
VAR_INPUT
    fScalar     : LREAL;
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fScalar	LREAL	Scalar value
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

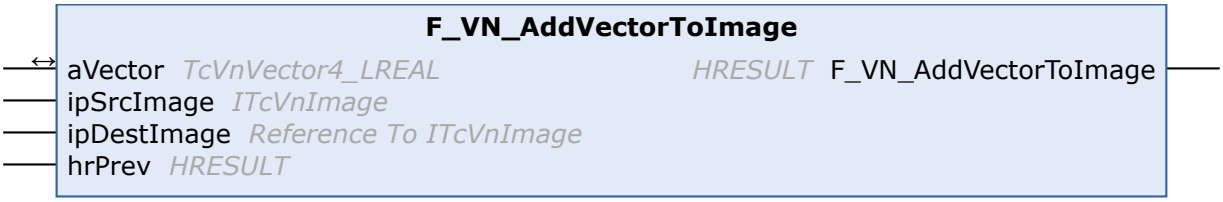
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.5 F_VN_AddVectorToImage



Add a vector (1 element for each image channel) to each image pixel using saturation arithmetics.

Syntax

Definition:

```
FUNCTION F_VN_AddVectorToImage : HRESULT
VAR_IN_OUT
    aVector     : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 173]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 150]

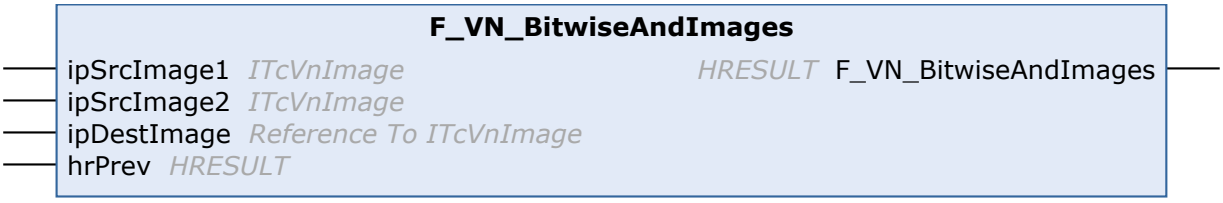
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.6 F_VN_BitwiseAndImages



Element-wise application of a bit-wise AND operator to two images.

Syntax

Definition:

```
FUNCTION F_VN_BitwiseAndImages : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
    ipSrcImage2 : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage1	ITcVnImage [▶ 426]	First source image
ipSrcImage2	ITcVnImage [▶ 426]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

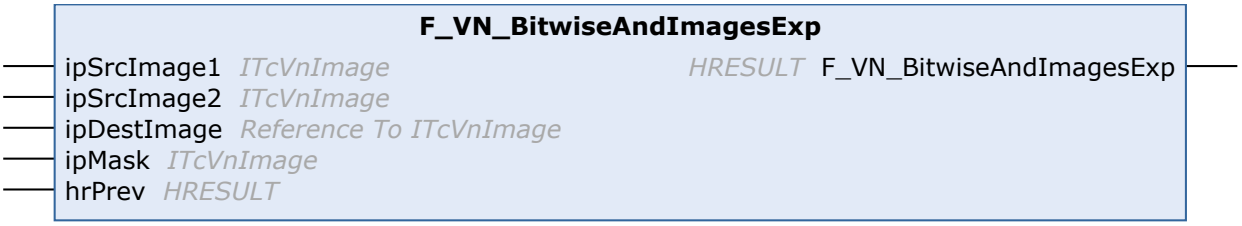
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.7 F_VN_BitwiseAndImagesExp



Element-wise application of a bit-wise AND operator to two images. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_BitwiseAndImagesExp : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
    ipSrcImage2 : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    ipMask      : ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage1	ITcVnImage [▶ 426]	First source image
ipSrcImage2	ITcVnImage [▶ 426]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage [▶ 426]	Mask to limit the operation to specific pixel positions (mask value 0: skip pixel, mask value > 0: apply operation to pixel)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

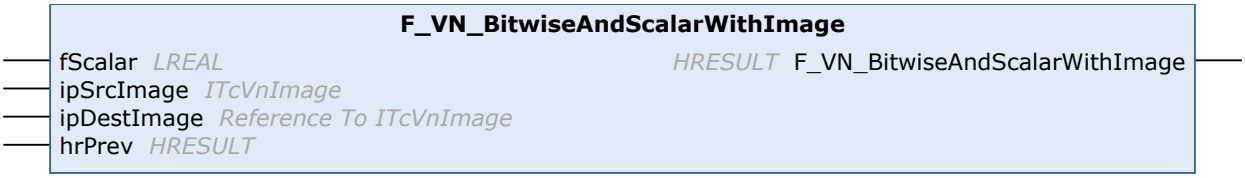
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.8 F_VN_BitwiseAndScalarWithImage



Bitwise and a scalar value with each image pixel.

Syntax

Definition:

```
FUNCTION F_VN_BitwiseAndScalarWithImage : HRESULT
VAR_INPUT
    fScalar     : LREAL;
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fScalar	LREAL	Scalar value
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

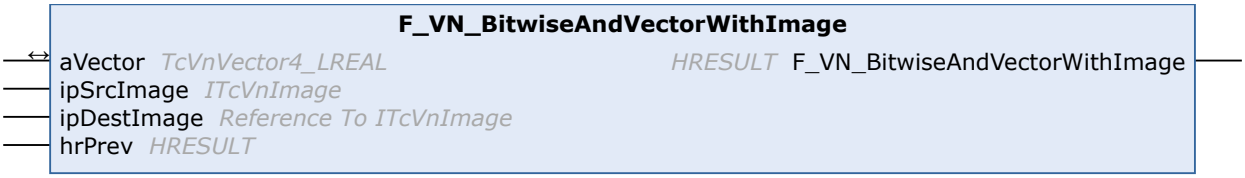
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.9 F_VN_BitwiseAndVectorWithImage



Bitwise and a vector (1 element for each image channel) with each image pixel.

Syntax

Definition:

```
FUNCTION F_VN_BitwiseAndVectorWithImage : HRESULT
VAR_IN_OUT
    aVector     : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 173]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 150]

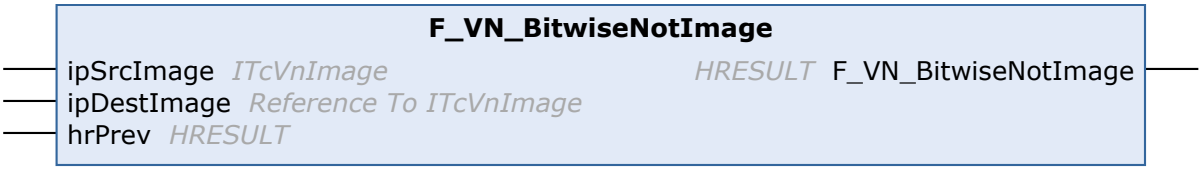
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.10 F_VN_BitwiseNotImage



Element-wise application of a bit-wise NOT operator to an image.

Syntax

Definition:

```
FUNCTION F_VN_BitwiseNotImage : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

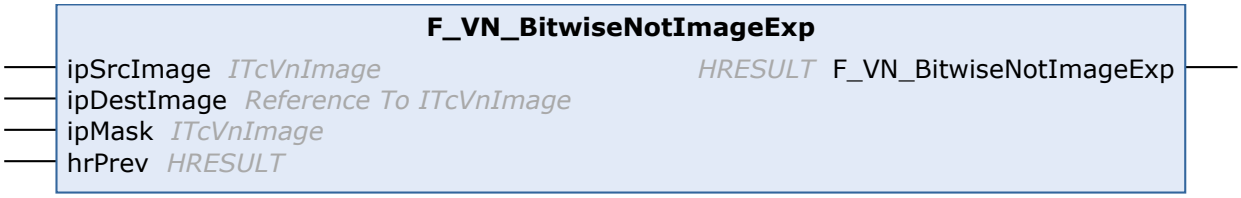
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.11 F_VN_BitwiseNotImageExp



Element-wise application of a bit-wise NOT operator to an image. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_BitwiseNotImageExp : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    ipMask      : ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage [▶ 426]	Mask to limit the operation to specific pixel positions (mask value 0: skip pixel, mask value > 0: apply operation to pixel)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

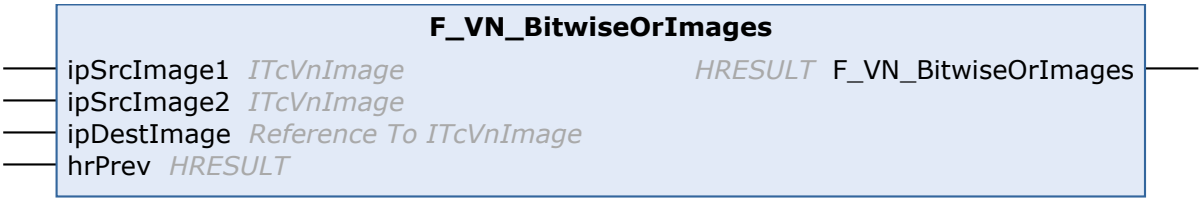
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.12 F_VN_BitwiseOrImages



Element-wise application of a bit-wise OR operator to two images.

Syntax

Definition:

```
FUNCTION F_VN_BitwiseOrImages : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
    ipSrcImage2 : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage1	ITcVnImage [▶ 426]	First source image
ipSrcImage2	ITcVnImage [▶ 426]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

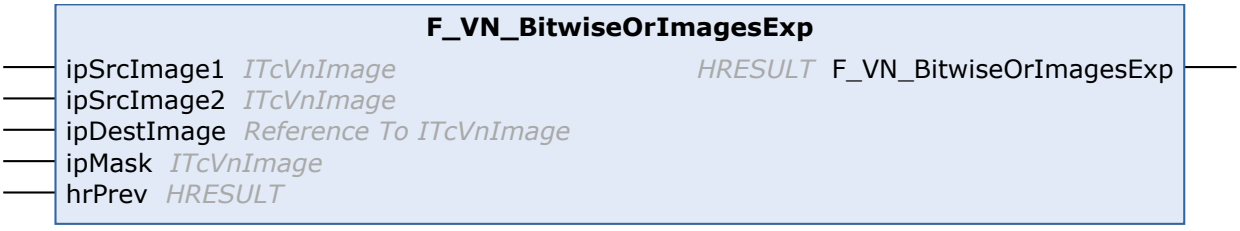
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.13 F_VN_BitwiseOrImagesExp



Element-wise application of a bit-wise OR operator to two images. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_BitwiseOrImagesExp : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
    ipSrcImage2 : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    ipMask      : ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage1	ITcVnImage [▶ 426]	First source image
ipSrcImage2	ITcVnImage [▶ 426]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage [▶ 426]	Mask to limit the operation to specific pixel positions (mask value 0: skip pixel, mask value > 0: apply operation to pixel)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

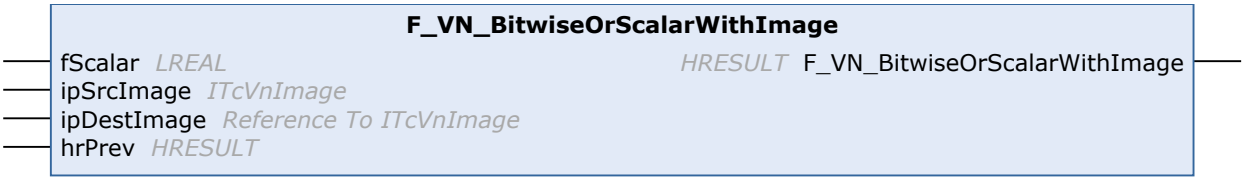
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.14 F_VN_BitwiseOrScalarWithImage



Bitwise or a scalar value with each image pixel.

Syntax

Definition:

```
FUNCTION F_VN_BitwiseOrScalarWithImage : HRESULT
VAR_INPUT
    fScalar     : LREAL;
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fScalar	LREAL	Scalar value
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

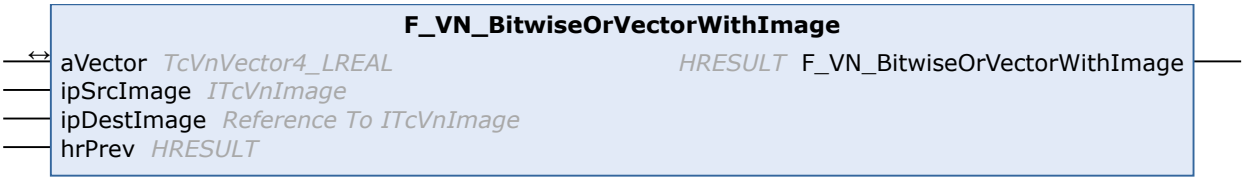
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.15 F_VN_BitwiseOrVectorWithImage



Bitwise or a vector (1 element for each image channel) with each image pixel.

Syntax

Definition:

```
FUNCTION F_VN_BitwiseOrVectorWithImage : HRESULT
VAR_IN_OUT
    aVector     : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 173]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 150]

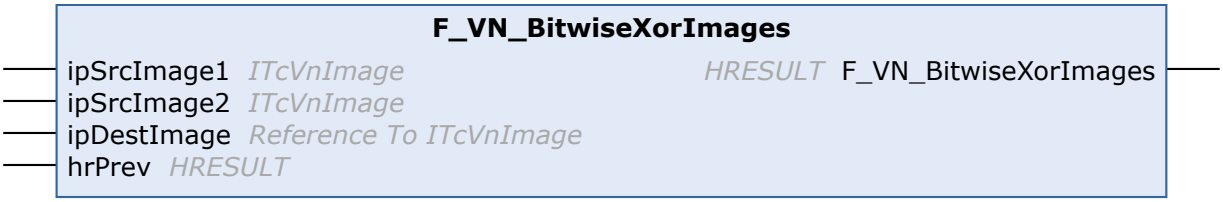
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.16 F_VN_BitwiseXorImages



Element-wise application of a bit-wise XOR operator to two images.

Syntax

Definition:

```
FUNCTION F_VN_BitwiseXorImages : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
    ipSrcImage2 : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage1	ITcVnImage [▶ 426]	First source image
ipSrcImage2	ITcVnImage [▶ 426]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

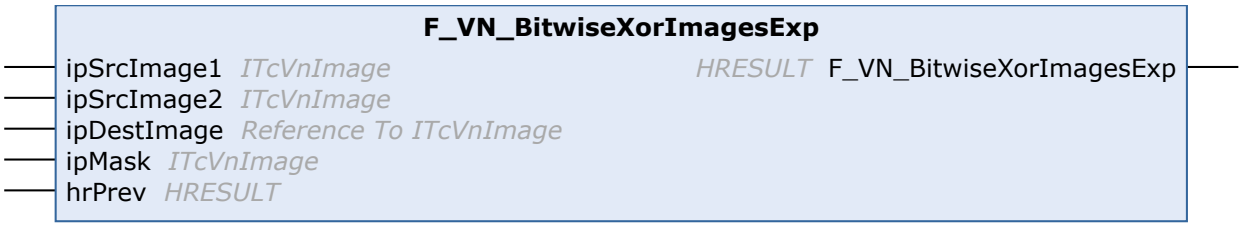
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.17 F_VN_BitwiseXorImagesExp



Element-wise application of a bit-wise XOR operator to two images. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_BitwiseXorImagesExp : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
    ipSrcImage2 : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    ipMask      : ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage1	ITcVnImage [▶ 426]	First source image
ipSrcImage2	ITcVnImage [▶ 426]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage [▶ 426]	Mask to limit the operation to specific pixel positions (mask value 0: skip pixel, mask value > 0: apply operation to pixel)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

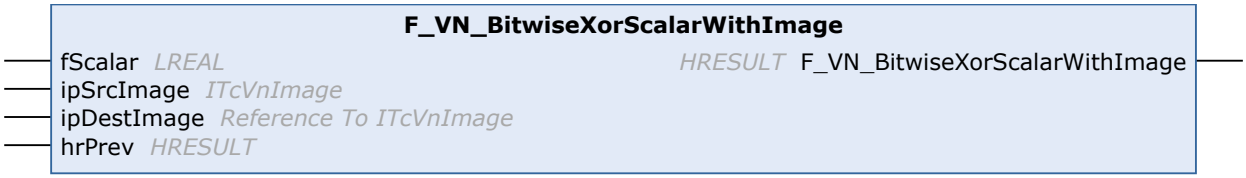
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.18 F_VN_BitwiseXorScalarWithImage



Bitwise xor a scalar value with each image pixel.

Syntax

Definition:

```
FUNCTION F_VN_BitwiseXorScalarWithImage : HRESULT
VAR_INPUT
    fScalar     : LREAL;
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fScalar	LREAL	Scalar value
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

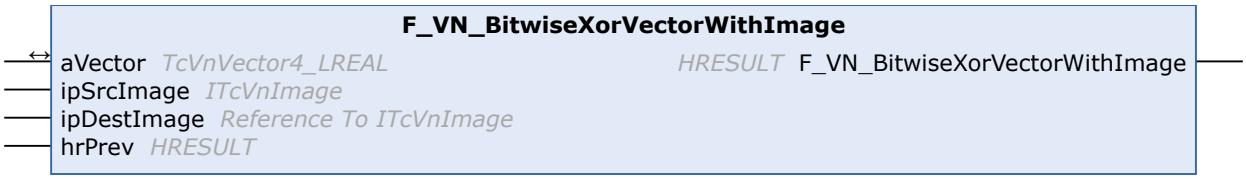
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.19 F_VN_BitwiseXorVectorWithImage



Bitwise xor a vector (1 element for each image channel) with each image pixel.

Syntax

Definition:

```
FUNCTION F_VN_BitwiseXorVectorWithImage : HRESULT
VAR_IN_OUT
    aVector     : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 173]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 150]

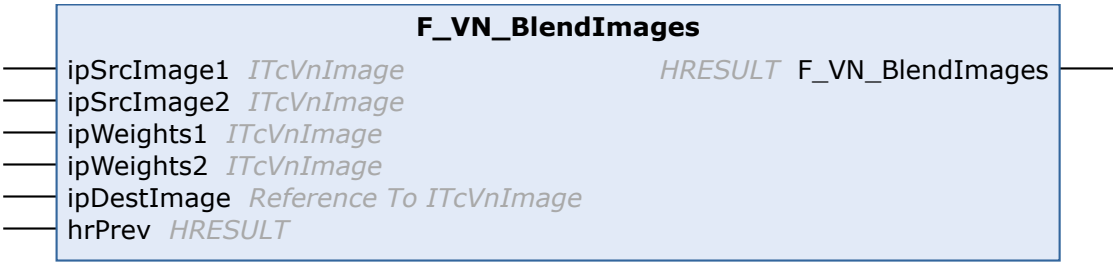
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.20 F_VN_BlendImages



Blends two images, i.e. dest = (src1 * weight1 + src2 * weight2) / (weight1 + weight2).
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_BlendImages : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
```

```
    ipSrcImage2 : ITcVnImage;
    ipWeights1  : ITcVnImage;
    ipWeights2  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage1	ITcVnImage [▶ 426]	First source image (ET_USINT or ET_REAL)
ipSrcImage2	ITcVnImage [▶ 426]	Second source image (same type and size as ipSrcImage1)
ipWeights1	ITcVnImage [▶ 426]	Weights for ipSrcImage1 (1 channel, ET_REAL)
ipWeights2	ITcVnImage [▶ 426]	Weights for ipSrcImage2 (1 channel, ET_REAL)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

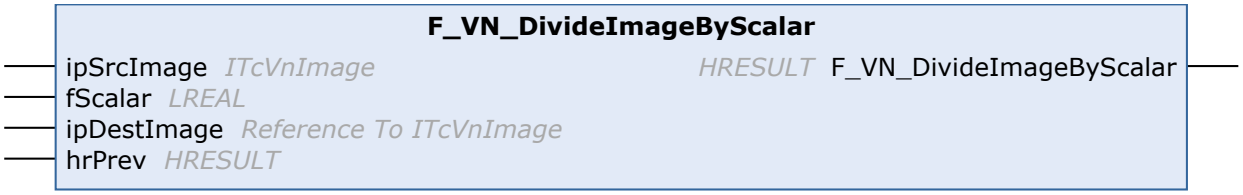
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.21 F_VN_DivideImageByScalar



Divide each image pixel by a scalar value using saturation arithmetics.

Syntax

Definition:

```
FUNCTION F_VN_DivideImageByScalar : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    fScalar     : LREAL;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
fScalar	LREAL	Scalar value
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

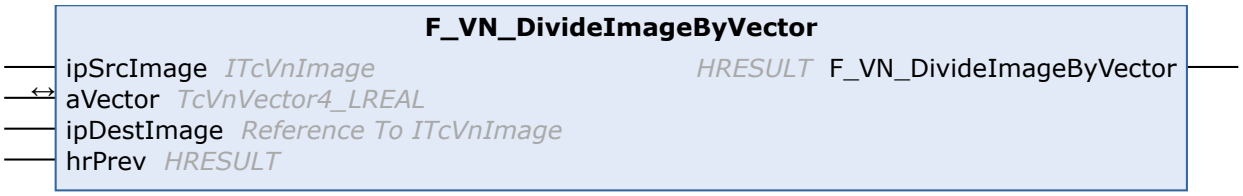
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.22 F_VN_DivideImageByVector



Divides each image pixel by a vector (1 element for each image channel) using saturation arithmetics.

Syntax

Definition:

```
FUNCTION F_VN_DivideImageByVector : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
END_VAR
VAR_IN_OUT
    aVector     : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 173]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.23 F_VN_DivideImages

Element-wise division of two images using saturation arithmetics. (A division by zero equals zero.)

Syntax

Definition:

```
FUNCTION F_VN_DivideImages : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
    ipSrcImage2 : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage1	ITcVnImage [▶ 426]	First source image (dividend)
ipSrcImage2	ITcVnImage [▶ 426]	Second source image (divisor)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_DivideImages` gibt bei Divisionen durch Null für die jeweiligen Elemente Null zurück. Dies funktioniert allerdings nur, wenn die Option **Floating point exceptions** der ausführenden SPS-Task [▶ 63] deaktiviert ist. Anderenfalls tritt ein Division-by-zero Fehler auf.

HINWEIS

Floating Point Exceptions

Diese Funktion kann Fehler herbeiführen, wenn die Option **Floating point exceptions** der ausführenden SPS-Task [▶ 63] aktiv ist. Deaktivieren Sie diese Option deshalb.

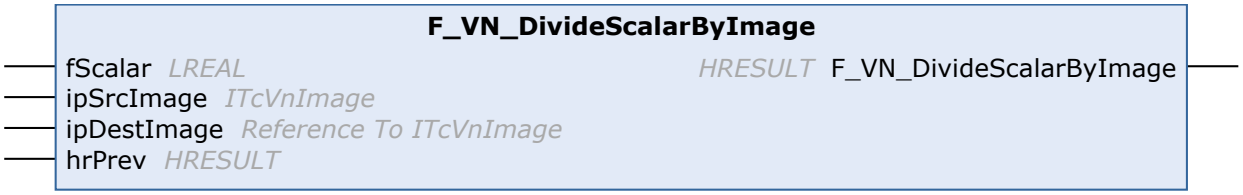
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.24 F_VN_DivideScalarByImage



Divides a scalar value by each image pixel using saturation arithmetics.

Syntax

Definition:

```
FUNCTION F_VN_DivideScalarByImage : HRESULT
VAR_INPUT
    fScalar     : LREAL;
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fScalar	LREAL	Scalar value
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

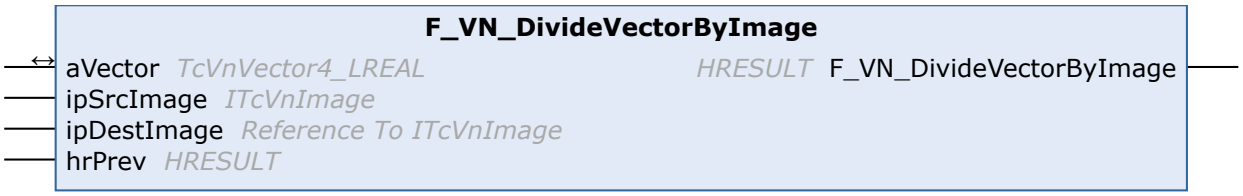
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.25 F_VN_DivideVectorByImage



Divides a vector (1 element for each image channel) by each image pixel using saturation arithmetics.

Syntax

Definition:

```
FUNCTION F_VN_DivideVectorByImage : HRESULT
VAR_IN_OUT
    aVector     : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 173]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 150]

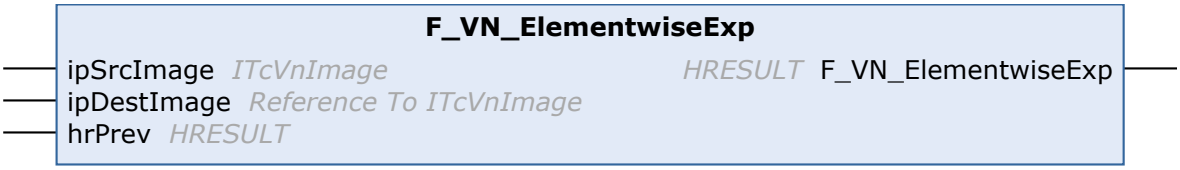
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.26 F_VN_ElementwiseExp



Computes the natural exponent of each pixel value.

Syntax

Definition:

```
FUNCTION F_VN_ElementwiseExp : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (REAL or LREAL)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (Same type and size as ipSrcImage, an appropriate image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.27 F_VN_ElementwiseLog

Computes the natural logarithm of each pixel value.

Syntax

Definition:

```
FUNCTION F_VN_ElementwiseLog : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (REAL or LREAL)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (Same type and size as ipSrcImage, an appropriate image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.28 F_VN_MaxImage

Element-wise maximum of two images.

Syntax

Definition:

```
FUNCTION F_VN_MaxImage : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
    ipSrcImage2 : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage1	ITcVnImage [▶ 426]	First source image
ipSrcImage2	ITcVnImage [▶ 426]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.29 F_VN_MaxImageExp

Element-wise maximum of two images. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_MaxImageExp : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
    ipSrcImage2 : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    ipMask      : ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage1	ITcVnImage [▶ 426]	First source image
ipSrcImage2	ITcVnImage [▶ 426]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage [▶ 426]	Mask of type USINT
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

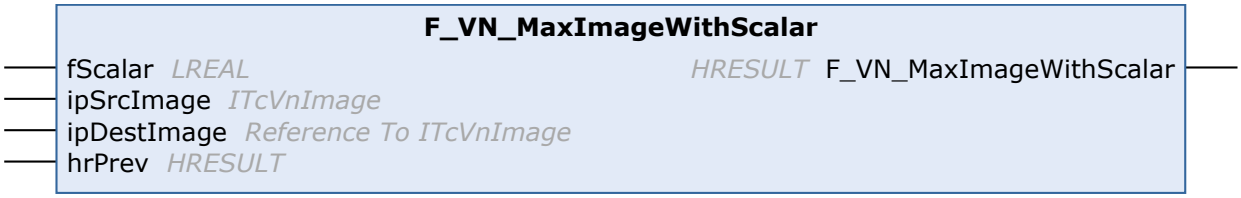
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.30 F_VN_MaxImageWithScalar



Element-wise maximum of image and scalar value.

Syntax

Definition:

```
FUNCTION F_VN_MaxImageWithScalar : HRESULT
VAR_INPUT
    fScalar     : LREAL;
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fScalar	LREAL	Scalar value
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

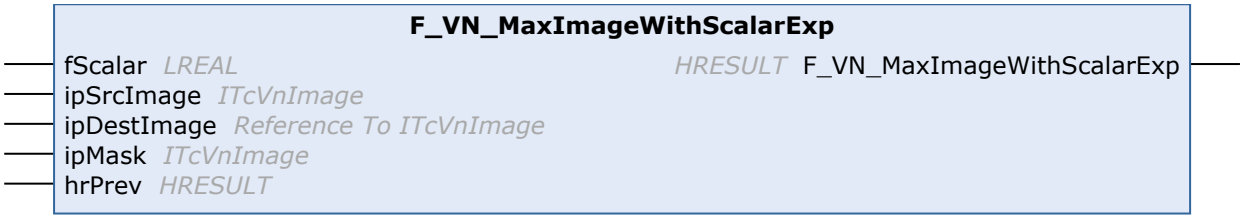
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.31 F_VN_MaxImageWithScalarExp



Element-wise maximum of image and scalar value. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_MaxImageWithScalarExp : HRESULT
VAR_INPUT
    fScalar     : LREAL;
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    ipMask      : ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fScalar	LREAL	Scalar value
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage [▶ 426]	Mask of type USINT
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

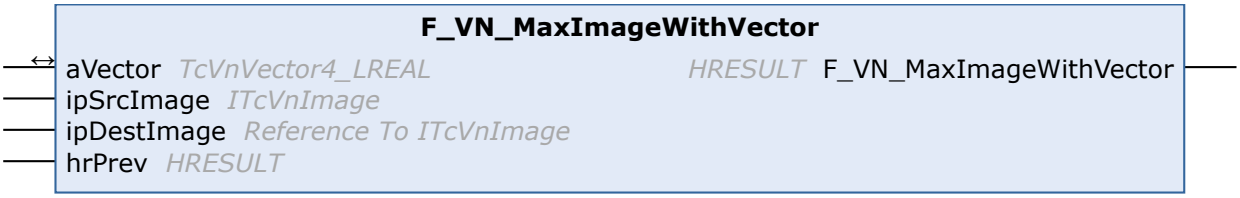
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.32 F_VN_MaxImageWithVector



Element-wise maximum of image and vector (1 element for each image channel).

Syntax

Definition:

```
FUNCTION F_VN_MaxImageWithVector : HRESULT
VAR_IN_OUT
    aVector     : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 173]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 150]

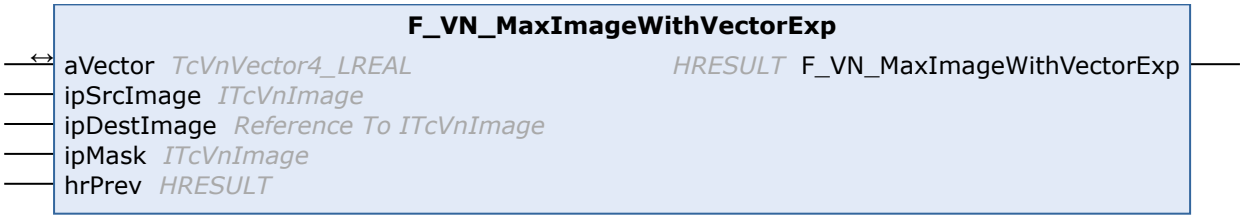
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.33 F_VN_MaxImageWithVectorExp



Element-wise maximum of image and vector (1 element for each image channel). (expert function)

Syntax

Definition:

```
FUNCTION F_VN_MaxImageWithVectorExp : HRESULT
VAR_IN_OUT
    aVector     : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    ipMask      : ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage [▶ 426]	Mask of type USINT
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 173]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.34 F_VN_MinImage

Element-wise minimum of two images.

Syntax

Definition:

```
FUNCTION F_VN_MinImage : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
    ipSrcImage2 : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage1	ITcVnImage [▶ 426]	First source image
ipSrcImage2	ITcVnImage [▶ 426]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.35 F_VN_MinImageExp

Element-wise minimum of two images. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_MinImageExp : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
    ipSrcImage2 : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    ipMask      : ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage1	ITcVnImage [▶ 426]	First source image
ipSrcImage2	ITcVnImage [▶ 426]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage [▶ 426]	Mask of type USINT
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.36 F_VN_MinImageWithScalar

Element-wise minimum of image and scalar value.

Syntax

Definition:

```
FUNCTION F_VN_MinImageWithScalar : HRESULT
VAR_INPUT
    fScalar     : LREAL;
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fScalar	LREAL	Scalar value
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

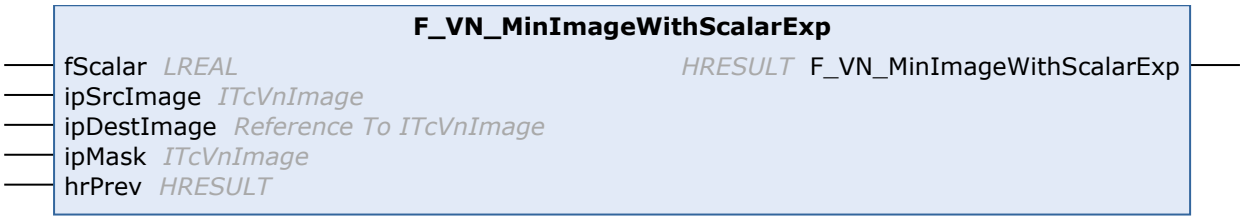
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.37 F_VN_MinImageWithScalarExp



Element-wise minimum of image and scalar value. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_MinImageWithScalarExp : HRESULT
VAR_INPUT
    fScalar     : LREAL;
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    ipMask      : ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fScalar	LREAL	Scalar value
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage [▶ 426]	Mask of type USINT
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.38 F_VN_MinImageWithVector

Element-wise minimum of image and vector (1 element for each image channel).

Syntax

Definition:

```
FUNCTION F_VN_MinImageWithVector : HRESULT
VAR_IN_OUT
    aVector     : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 173]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 150]

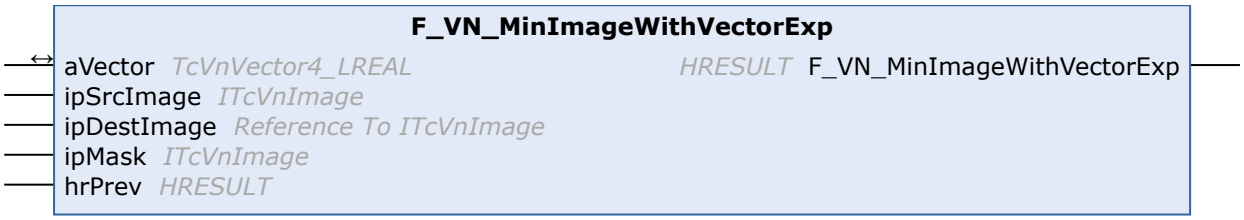
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.39 F_VN_MinImageWithVectorExp



Element-wise minimum of image and vector (1 element for each image channel). (expert function)

Syntax

Definition:

```
FUNCTION F_VN_MinImageWithVectorExp : HRESULT
VAR_IN_OUT
    aVector     : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    ipMask      : ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage [▶ 426]	Mask of type USINT
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 173]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.40 F_VN_MultiplyImages

Element-wise multiplication of two images using saturation arithmetics.

Syntax

Definition:

```
FUNCTION F_VN_MultiplyImages : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
    ipSrcImage2 : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage1	ITcVnImage [▶ 426]	First source image
ipSrcImage2	ITcVnImage [▶ 426]	Second source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

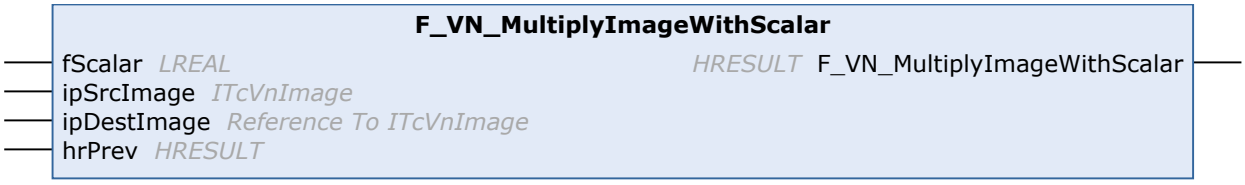
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.41 F_VN_MultiplyImageWithScalar



Multiply each image pixel by a scalar value using saturation arithmetics.

Syntax

Definition:

```
FUNCTION F_VN_MultiplyImageWithScalar : HRESULT
VAR_INPUT
    fScalar     : LREAL;
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fScalar	LREAL	Scalar value
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.42 F_VN_MultiplyImageWithVector

Multiply each image pixel by a vector (1 element for each image channel) using saturation arithmetics.

Syntax

Definition:

```
FUNCTION F_VN_MultiplyImageWithVector : HRESULT
VAR_IN_OUT
    aVector     : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 173]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 150]

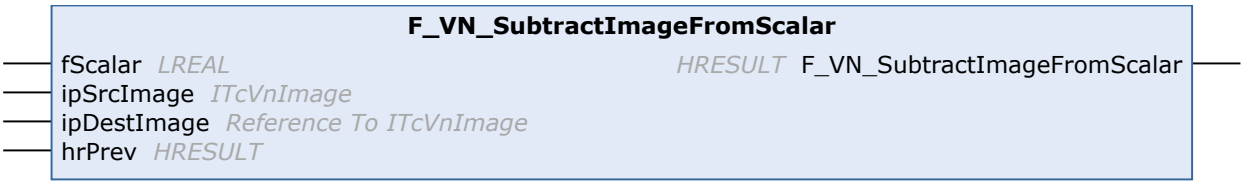
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.43 F_VN_SubtractImageFromScalar



Subtract each image pixel from a scalar value using saturation arithmetics.

Syntax

Definition:

```
FUNCTION F_VN_SubtractImageFromScalar : HRESULT
VAR_INPUT
    fScalar     : LREAL;
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fScalar	LREAL	Scalar value
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

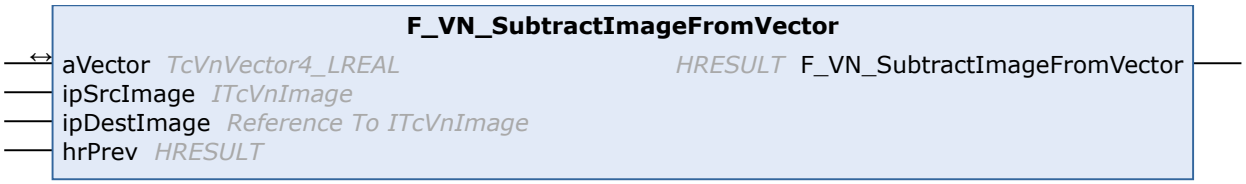
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.44 F_VN_SubtractImageFromVector



Subtract each image pixel from a vector (1 element for each image channel) using saturation arithmetics.

Syntax

Definition:

```
FUNCTION F_VN_SubtractImageFromVector : HRESULT
VAR_IN_OUT
    aVector     : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 173]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 150]

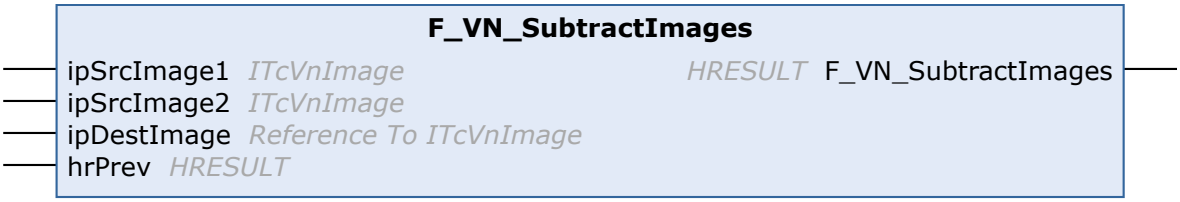
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.45 F_VN_SubtractImages



Element-wise subtraction of two images using saturation arithmetics.

Syntax

Definition:

```
FUNCTION F_VN_SubtractImages : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
    ipSrcImage2 : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage1	ITcVnImage [▶ 426]	First source image (minuend)
ipSrcImage2	ITcVnImage [▶ 426]	Second source image (subtrahend)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

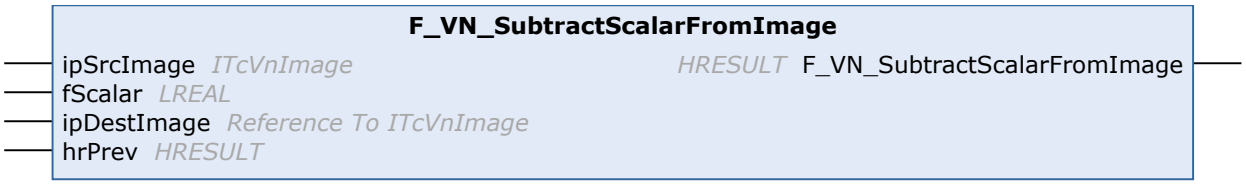
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.46 F_VN_SubtractScalarFromImage



Subtract a scalar value from each image pixel using saturation arithmetics.

Syntax

Definition:

```
FUNCTION F_VN_SubtractScalarFromImage : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    fScalar     : LREAL;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
fScalar	LREAL	Scalar value
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

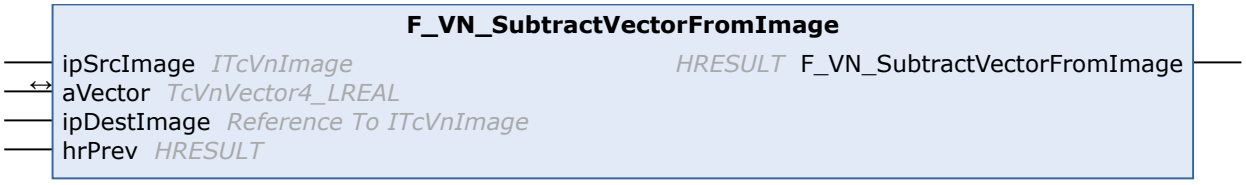
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.3.47 F_VN_SubtractVectorFromImage



Subtract a vector (1 element for each image channel) from each image pixel using saturation arithmetics.

Syntax

Definition:

```
FUNCTION F_VN_SubtractVectorFromImage : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
END_VAR
VAR_IN_OUT
    aVector     : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 173]	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4 Basic Container Operations

Diese Gruppe enthält Funktionen zur Handhabung von Containern [▶ 163].

Funktionen

Erstellung von Containern

- F_VN_CopyContainer [▶ 772]
- F_VN_CopyContainerElementsConditional_ITcVnContainer [▶ 773]
- F_VN_CopyContainerElementsConditional_ITcVnForwardIterator [▶ 774]
- F_VN_CreateContainer [▶ 775]
- F_VN_CreateContainerFromArray [▶ 776]
- F_VN_ExtractContainerRange [▶ 785]
- F_VN_ReserveContainerMemory [▶ 793]

Handhabung von Container-Elementen

- F_VN_AppendToContainer [▶ 566]
- F_VN_CheckIfEmpty [▶ 769]
- F_VN_ConvertContainerType [▶ 771]
- F_VN_EraseFromContainer [▶ 777]
- F_VN_FillContainer(Exp) [▶ 627]
- F_VN_GetAt [▶ 663]
- F_VN_GetContainer(Exp) [▶ 786]
- F_VN_GetNumberOfElements [▶ 789]
- F_VN_InsertIntoContainer [▶ 699]
- F_VN_ReverseContainer [▶ 793]
- F_VN_SetAt [▶ 732]
- F_VN_SetContainer [▶ 794]

Container-Daten in Speicher-Bereich exportieren

- F_VN_ExportContainer [▶ 778]
- F_VN_ExportSubContainer [▶ 782]
- F_VN_ExportContainer_String [▶ 780]
- F_VN_ExportSubContainer_String [▶ 783]
- F_VN_ExportContainerSize [▶ 781]
- F_VN_ExportSubContainerSize [▶ 784]

Handhabung von Iteratoren

- F_VN_AdvanceIterator [▶ 768]
- F_VN_CheckIfIteratorIsAtEnd [▶ 770]
- F_VN_GetForwardIterator [▶ 788]
- F_VN_GetRandomAccessIterator [▶ 790]
- F_VN_IncrementIterator [▶ 791]
- F_VN_IteratorDistance [▶ 792]
- F_VN_SetIteratorToBegin [▶ 795]

Beispiele

Basic Container Operations [▶ 2840]

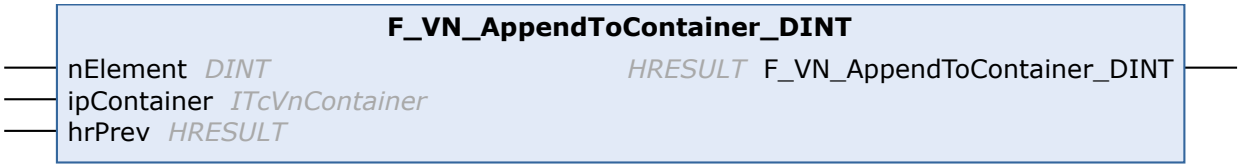
Sehen Sie dazu auch

F_VN_PadImageBorder [▶ 841]

6.1.4.4.1 F_VN_AppendToContainer

Dieses Kapitel enthält Funktionen, mit denen ein Element zu einem Container entsprechend dem Datentyp hinzugefügt werden kann.

6.1.4.4.1.1 F_VN_AppendToContainer_DINT



Append a single element of type DINT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_DINT : HRESULT
VAR_INPUT
    nElement    : DINT;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nElement	DINT	Single element to append to ipContainer
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

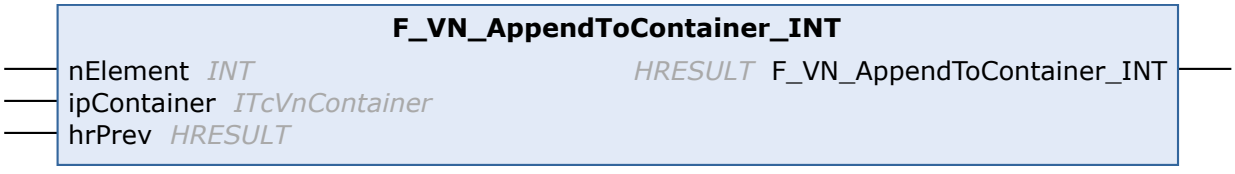
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.2 F_VN_AppendToContainer_INT



Append a single element of type INT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_INT : HRESULT
VAR_INPUT
    nElement    : INT;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nElement	INT	Single element to append to ipContainer
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

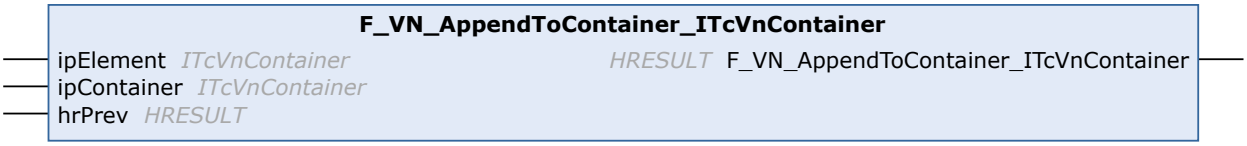
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.3 F_VN_AppendToContainer_ITcVnContainer



Append a single element to a container or concatenate containers (if ipElement has the same typeId as ipContainer).

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_ITcVnContainer : HRESULT
VAR_INPUT
    ipElement   : ITcVnContainer;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipElement	ITcVnContainer [▶ 385]	Single element to append to ipContainer or container with several elements to be concatenated
ipContainer	ITcVnContainer [▶ 385]	Container to which the element(s) will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

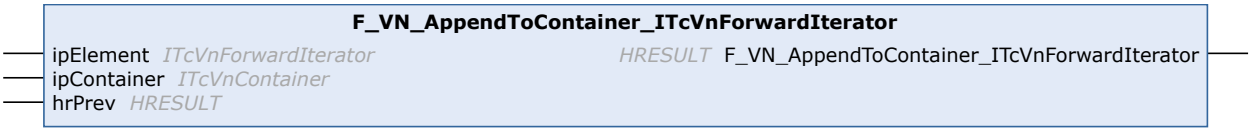
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.4 F_VN_AppendToContainer_ITcVnForwardIterator



Append a single element (represented by an iterator)to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_ITcVnForwardIterator : HRESULT
VAR_INPUT
    ipElement   : ITcVnForwardIterator;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipElement	ITcVnForwardIterator [▶ 376]	Single element to append to ipContainer
ipContainer	ITcVnContainer [▶ 385]	Container to which the element(s) will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

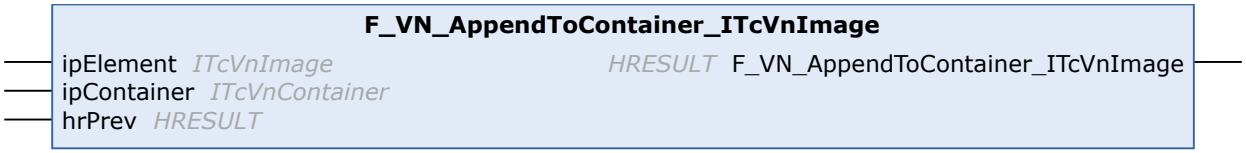
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.5 F_VN_AppendToContainer_ITcVnImage



Append a single element of type ITcVnImage to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_ITcVnImage : HRESULT
VAR_INPUT
    ipElement   : ITcVnImage;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipElement	ITcVnImage [▶ 426]	Single element to append to ipContainer
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

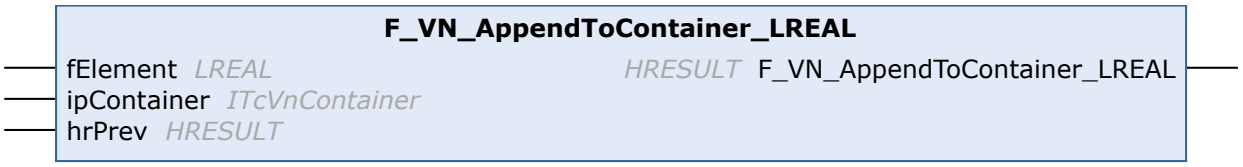
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.6 F_VN_AppendToContainer_LREAL



Append a single element of type LREAL to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_LREAL : HRESULT
VAR_INPUT
    fElement    : LREAL;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fElement	LREAL	Single element to append to ipContainer
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

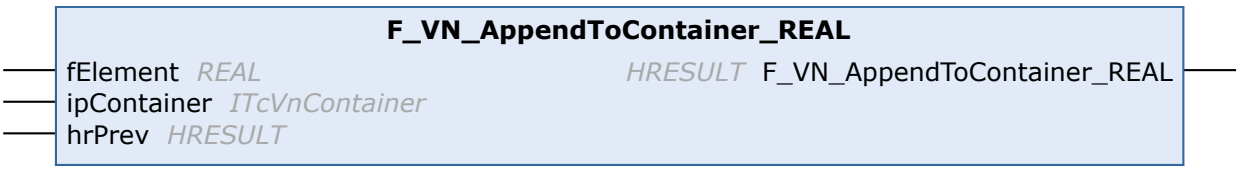
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.7 F_VN_AppendToContainer_REAL



Append a single element of type REAL to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_REAL : HRESULT
VAR_INPUT
    fElement    : REAL;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fElement	REAL	Single element to append to ipContainer
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

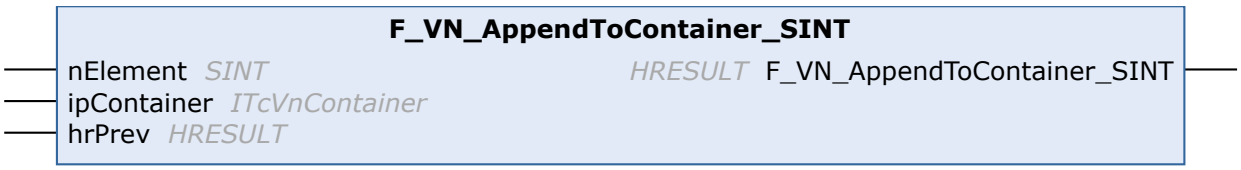
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.8 F_VN_AppendToContainer_SINT



Append a single element of type SINT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_SINT : HRESULT
VAR_INPUT
    nElement    : SINT;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nElement	SINT	Single element to append to ipContainer
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

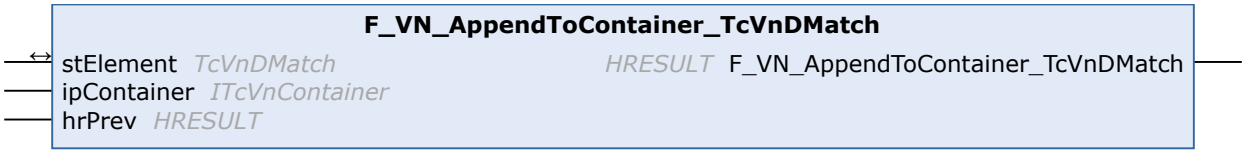
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.9 F_VN_AppendToContainer_TcVnDMatch



Append a single element of type TcVnDMatch to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnDMatch : HRESULT
VAR_IN_OUT
    stElement   : TcVnDMatch;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnDMatch [▶ 244]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

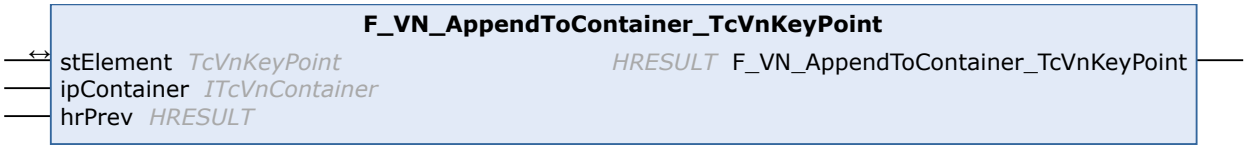
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.10 F_VN_AppendToContainer_TcVnKeyPoint



Append a single element of type TcVnKeyPoint to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnKeyPoint : HRESULT
VAR_IN_OUT
    stElement   : TcVnKeyPoint;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnKeyPoint [▶ 244]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

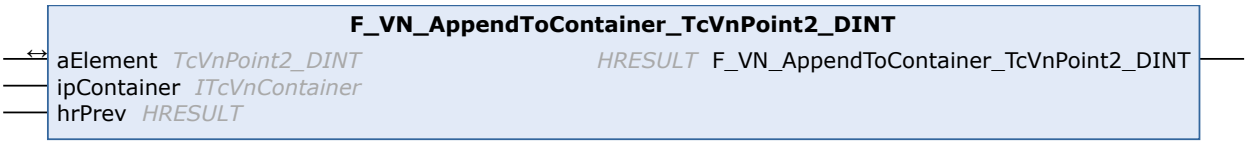
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.11 F_VN_AppendToContainer_TcVnPoint2_DINT



Append a single element of type TcVnPoint2_DINT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnPoint2_DINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnPoint2_DINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_DINT [▶ 171]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

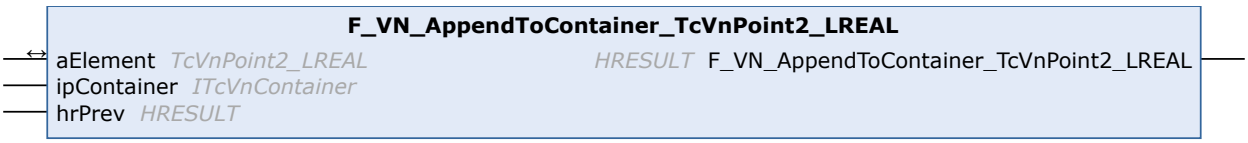
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.12 F_VN_AppendToContainer_TcVnPoint2_LREAL



Append a single element of type TcVnPoint2_LREAL to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnPoint2_LREAL : HRESULT
VAR_IN_OUT
    aElement    : TcVnPoint2_LREAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_LREAL [▶ 171]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

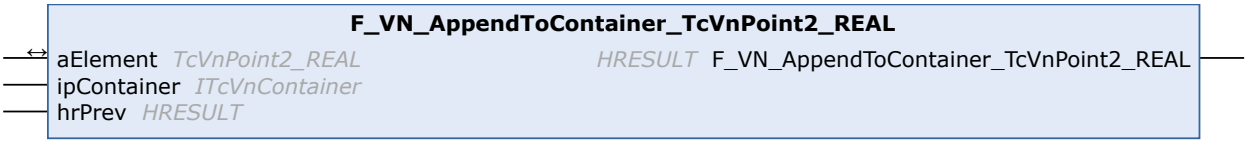
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.13 F_VN_AppendToContainer_TcVnPoint2_REAL



Append a single element of type TcVnPoint2_REAL to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnPoint2_REAL : HRESULT
VAR_IN_OUT
    aElement    : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_REAL [▶ 171]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

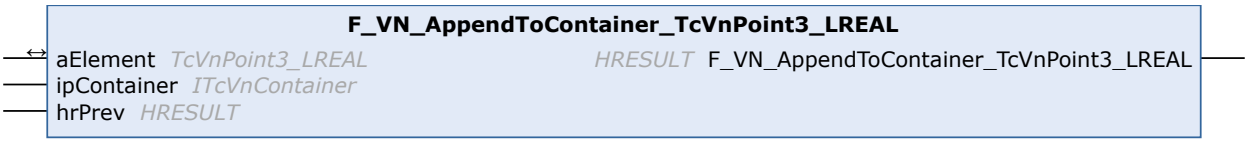
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.14 F_VN_AppendToContainer_TcVnPoint3_LREAL



Append a single element of type TcVnPoint3_LREAL to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnPoint3_LREAL : HRESULT
VAR_IN_OUT
    aElement    : TcVnPoint3_LREAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint3_LREAL [▶ 171]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

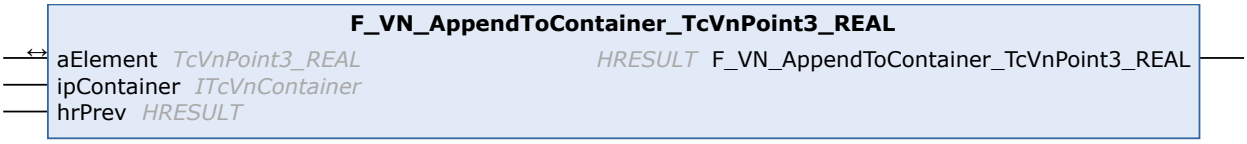
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.15 F_VN_AppendToContainer_TcVnPoint3_REAL



Append a single element of type TcVnPoint3_REAL to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnPoint3_REAL : HRESULT
VAR_IN_OUT
    aElement    : TcVnPoint3_REAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint3_REAL [▶ 171]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

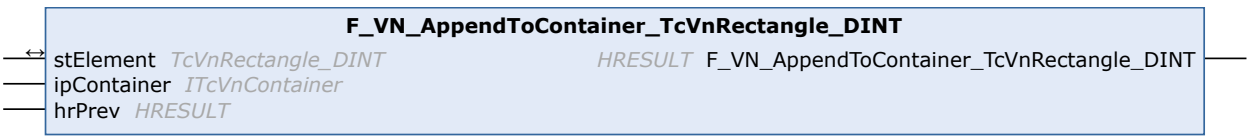
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.16 F_VN_AppendToContainer_TcVnRectangle_DINT



Append a single element of type TcVnRectangle_DINT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnRectangle_DINT : HRESULT
VAR_IN_OUT
    stElement   : TcVnRectangle_DINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnRectangle_DINT [▶ 257]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

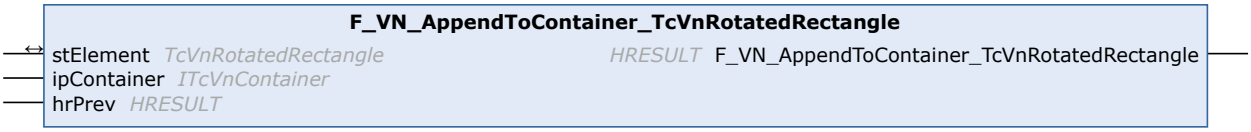
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.17 F_VN_AppendToContainer_TcVnRotatedRectangle



Append a single element of type TcVnRotatedRectangle to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnRotatedRectangle : HRESULT
VAR_IN_OUT
    stElement   : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnRotatedRectangle [▶ 258]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

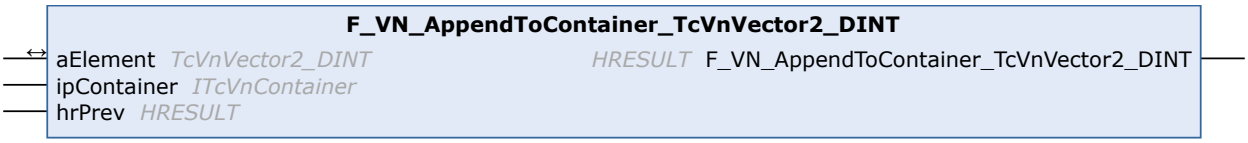
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.18 F_VN_AppendToContainer_TcVnVector2_DINT



Append a single element of type TcVnVector2_DINT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnVector2_DINT : HRESULT
VAR_IN_OUT
    aElement     : TcVnVector2_DINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_DINT [▶ 173]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

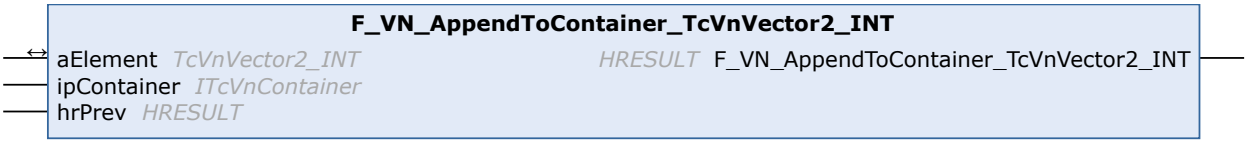
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.19 F_VN_AppendToContainer_TcVnVector2_INT



Append a single element of type TcVnVector2_INT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnVector2_INT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector2_INT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_INT [▶ 173]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

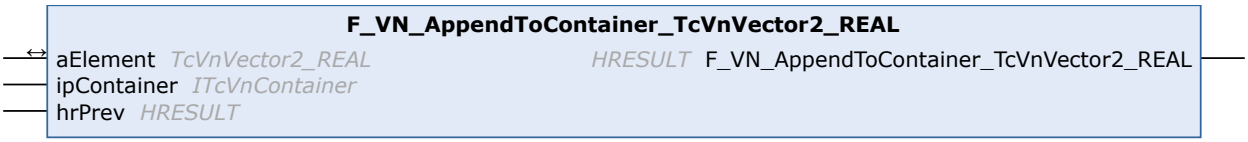
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.20 F_VN_AppendToContainer_TcVnVector2_REAL



Append a single element of type TcVnVector2_REAL to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnVector2_REAL : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector2_REAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_REAL [▶ 173]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

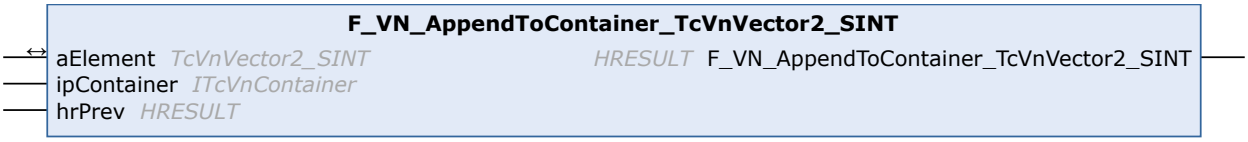
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.21 F_VN_AppendToContainer_TcVnVector2_SINT



Append a single element of type TcVnVector2_SINT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnVector2_SINT : HRESULT
VAR_IN_OUT
    aElement     : TcVnVector2_SINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_SINT [▶ 173]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

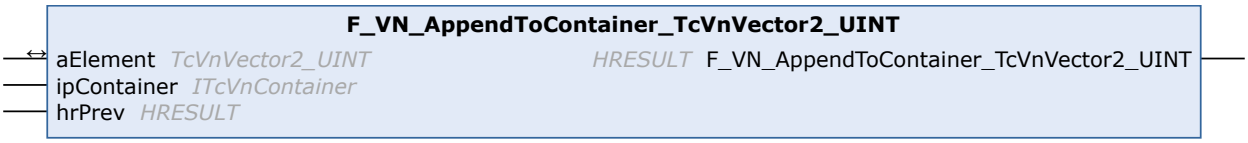
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.22 F_VN_AppendToContainer_TcVnVector2_UINT



Append a single element of type TcVnVector2_UINT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnVector2_UINT : HRESULT
VAR_IN_OUT
    aElement     : TcVnVector2_UINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_UINT [▶ 173]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

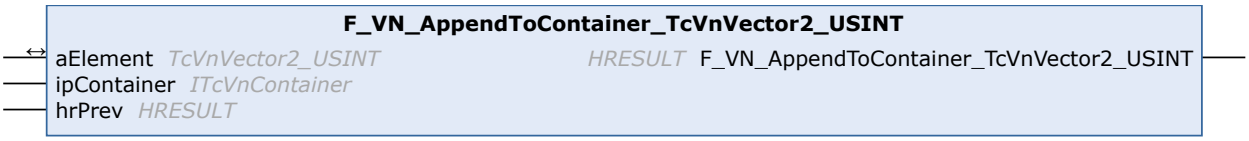
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.23 F_VN_AppendToContainer_TcVnVector2_USINT



Append a single element of type TcVnVector2_USINT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnVector2_USINT : HRESULT
VAR_IN_OUT
    aElement     : TcVnVector2_USINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_USINT [▶ 173]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

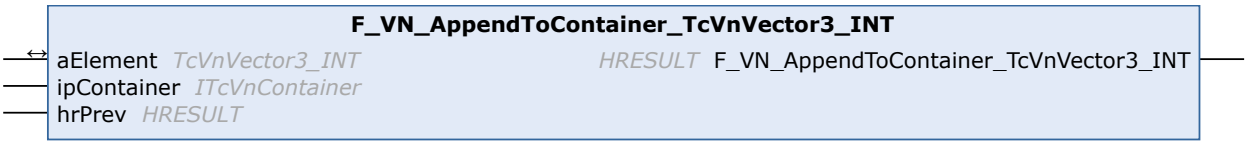
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.24 F_VN_AppendToContainer_TcVnVector3_INT



Append a single element of type TcVnVector3_INT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnVector3_INT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector3_INT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_INT [▶ 173]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

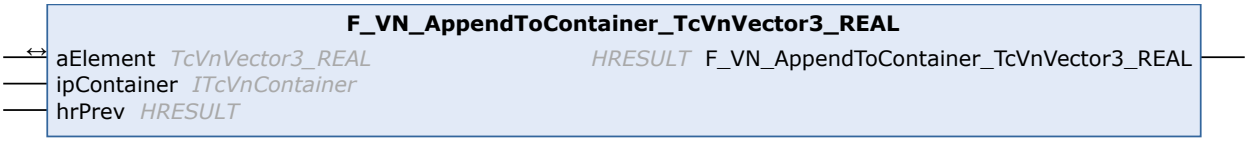
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.25 F_VN_AppendToContainer_TcVnVector3_REAL



Append a single element of type TcVnVector3_REAL to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnVector3_REAL : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector3_REAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_REAL [▶ 173]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

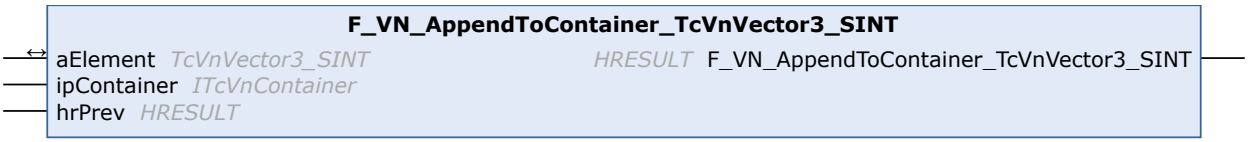
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.26 F_VN_AppendToContainer_TcVnVector3_SINT



Append a single element of type TcVnVector3_SINT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnVector3_SINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector3_SINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_SINT [▶ 173]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

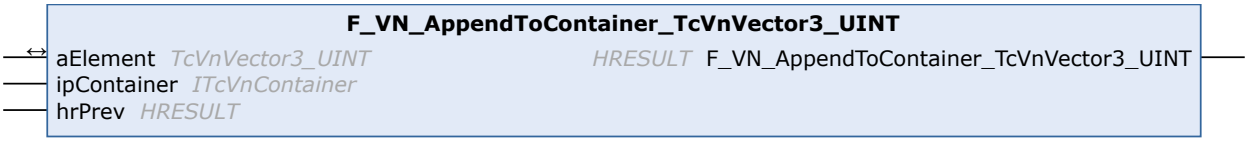
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.27 F_VN_AppendToContainer_TcVnVector3_UINT



Append a single element of type TcVnVector3_UINT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnVector3_UINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector3_UINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_UINT [▶ 173]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

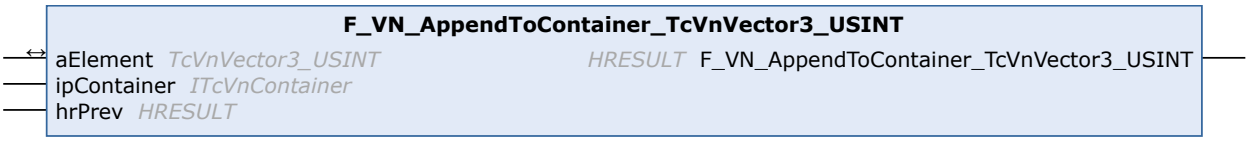
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.28 F_VN_AppendToContainer_TcVnVector3_USINT



Append a single element of type TcVnVector3_USINT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnVector3_USINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector3_USINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_USINT [▶ 173]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

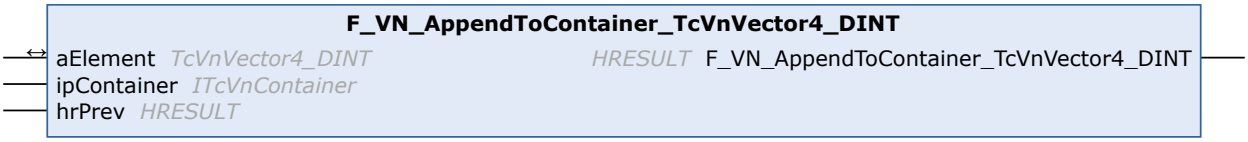
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.29 F_VN_AppendToContainer_TcVnVector4_DINT



Append a single element of type TcVnVector4_DINT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnVector4_DINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector4_DINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_DINT [▶ 173]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

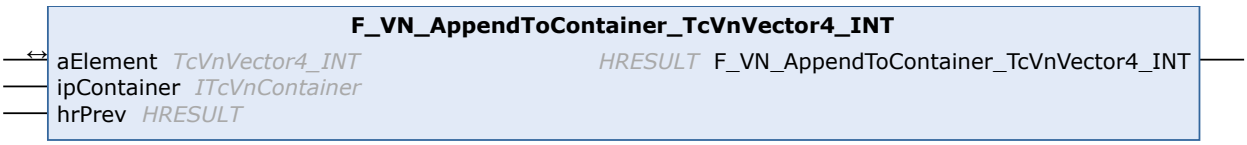
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.30 F_VN_AppendToContainer_TcVnVector4_INT



Append a single element of type TcVnVector4_INT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnVector4_INT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector4_INT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_INT [▶ 173]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.31 F_VN_AppendToContainer_TcVnVector4_LREAL

Append a single element of type TcVnVector4_LREAL to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnVector4_LREAL : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_LREAL [▶ 173]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

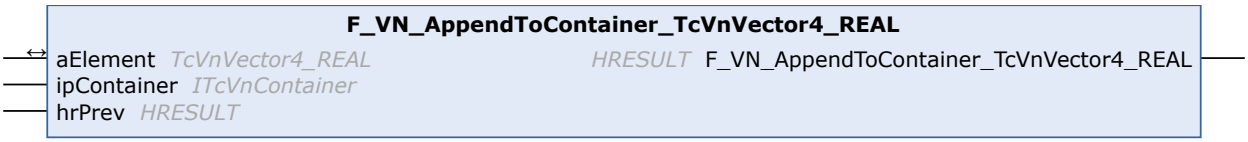
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.32 F_VN_AppendToContainer_TcVnVector4_REAL



Append a single element of type TcVnVector4_REAL to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnVector4_REAL : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector4_REAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_REAL [▶ 173]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

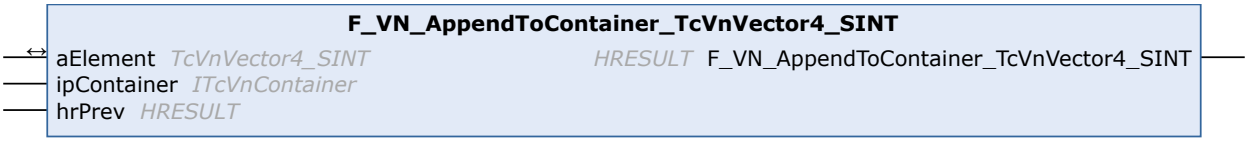
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.33 F_VN_AppendToContainer_TcVnVector4_SINT



Append a single element of type TcVnVector4_SINT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnVector4_SINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector4_SINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_SINT [▶ 173]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

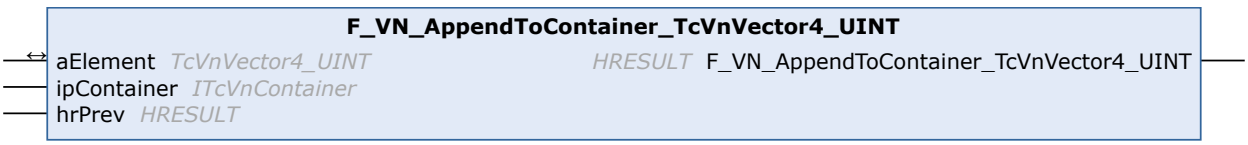
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.34 F_VN_AppendToContainer_TcVnVector4_UINT



Append a single element of type TcVnVector4_UINT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnVector4_UINT : HRESULT
VAR_IN_OUT
    aElement     : TcVnVector4_UINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_UINT [▶ 173]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

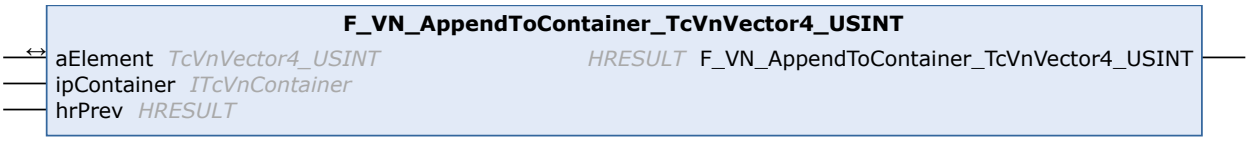
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.35 F_VN_AppendToContainer_TcVnVector4_USINT



Append a single element of type TcVnVector4_USINT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_TcVnVector4_USINT : HRESULT
VAR_IN_OUT
    aElement     : TcVnVector4_USINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_USINT [▶ 173]	Single element to append to ipContainer

Return value

HRESULT [▶ 150]

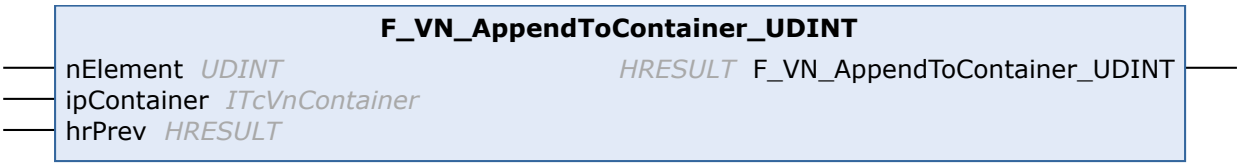
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.36 F_VN_AppendToContainer_UDINT



Append a single element of type UDINT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_UDINT : HRESULT
VAR_INPUT
    nElement    : UDINT;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nElement	UDINT	Single element to append to ipContainer
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

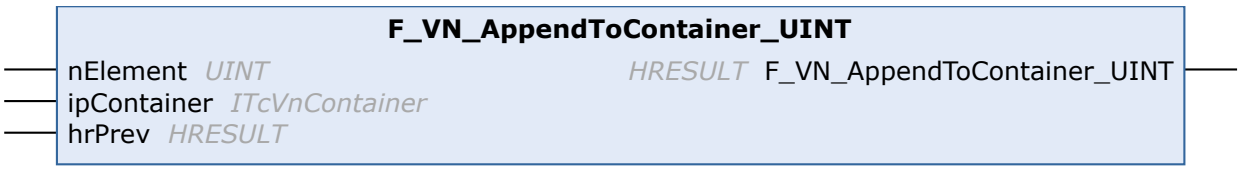
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.37 F_VN_AppendToContainer_UINT



Append a single element of type UINT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_UINT : HRESULT
VAR_INPUT
    nElement    : UINT;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nElement	UINT	Single element to append to ipContainer
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

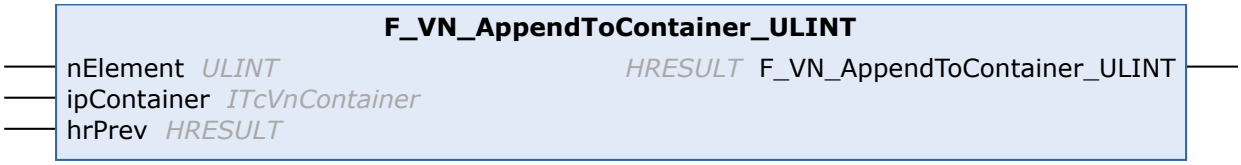
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.38 F_VN_AppendToContainer_ULINT



Append a single element of type ULINT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_ULINT : HRESULT
VAR_INPUT
    nElement    : ULINT;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nElement	ULINT	Single element to append to ipContainer
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

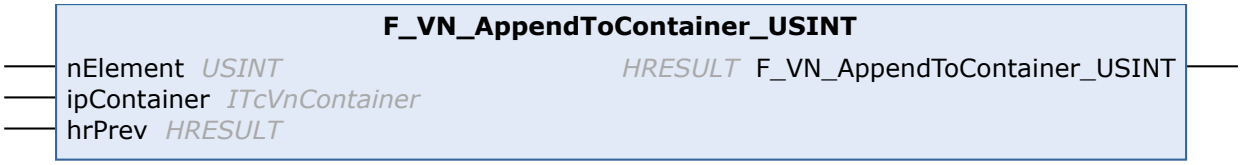
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.1.39 F_VN_AppendToContainer_USINT



Append a single element of type USINT to a container.

Syntax

Definition:

```
FUNCTION F_VN_AppendToContainer_USINT : HRESULT
VAR_INPUT
    nElement    : USINT;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nElement	USINT	Single element to append to ipContainer
ipContainer	ITcVnContainer [▶ 385]	Container to which the element will be appended
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2 F_VN_FillContainer

Dieses Kapitel enthält Funktionen, mit denen ein Container entsprechend dem Datentyp mit einem speziellen Wert befüllt werden kann. Diese Funktion ist für Container mit Grundelementen verfügbar.

6.1.4.4.2.1 F_VN_FillContainer_DINT

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nValue      : DINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nValue	DINT	Value to set the container elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

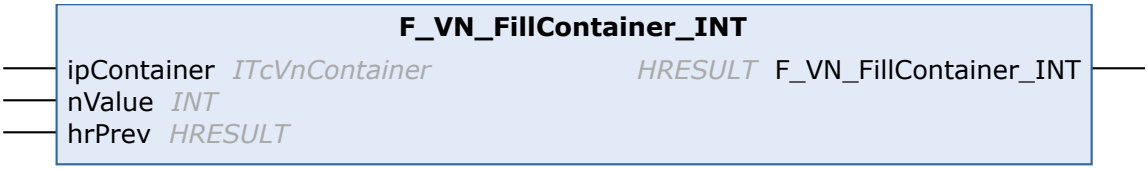
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.2 F_VN_FillContainer_INT



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nValue      : INT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nValue	INT	Value to set the container elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

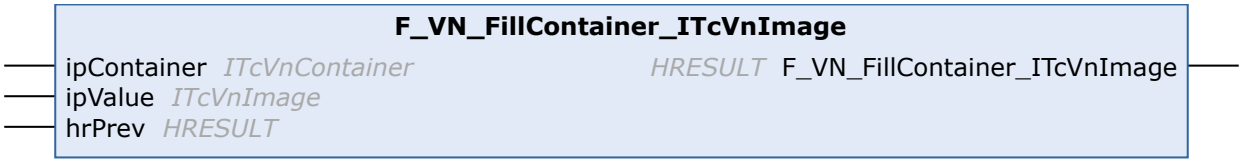
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.3 F_VN_FillContainer_ITcVnImage



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_ITcVnImage : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    ipValue     : ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
ipValue	ITcVnImage [▶ 426]	Value to set the container elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

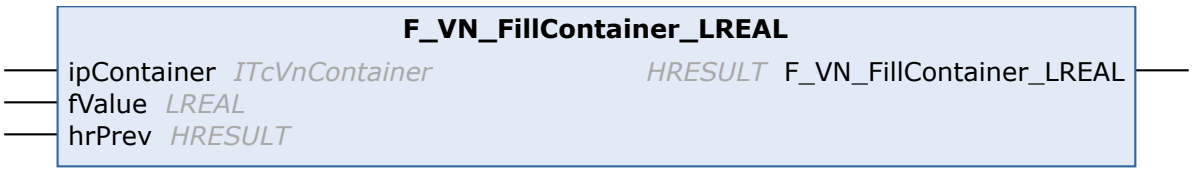
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.4 F_VN_FillContainer_LREAL



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    fValue      : LREAL;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
fValue	LREAL	Value to set the container elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.5 F_VN_FillContainer_REAL

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    fValue      : REAL;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
fValue	REAL	Value to set the container elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.6 F_VN_FillContainer_SINT

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nValue      : SINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nValue	SINT	Value to set the container elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

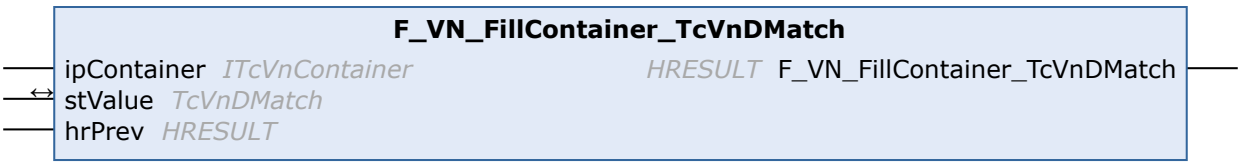
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.7 F_VN_FillContainer_TcVnDMatch



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnDMatch : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    stValue     : TcVnDMatch;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stValue	TcVnDMatch [▶ 244]	Value to set the container elements

Return value

HRESULT [▶ 150]

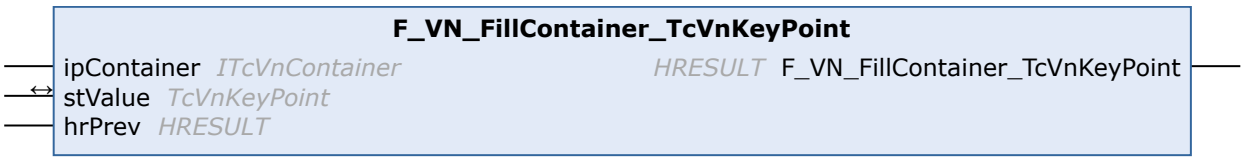
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.8 F_VN_FillContainer_TcVnKeyPoint



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnKeyPoint : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    stValue     : TcVnKeyPoint;
END_VAR
```

```
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stValue	TcVnKeyPoint [▶ 244]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.9 F_VN_FillContainer_TcVnPoint2_DINT

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnPoint2_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnPoint2_DINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint2_DINT [▶ 171]	Value to set the container elements

Return value

HRESULT [▶ 150]

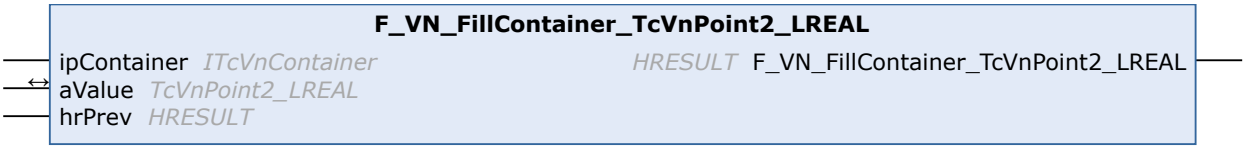
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.10 F_VN_FillContainer_TcVnPoint2_LREAL



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnPoint2_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnPoint2_LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint2_LREAL [▶ 171]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.11 F_VN_FillContainer_TcVnPoint2_REAL

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnPoint2_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint2_REAL [▶ 171]	Value to set the container elements

Return value

HRESULT [▶ 150]

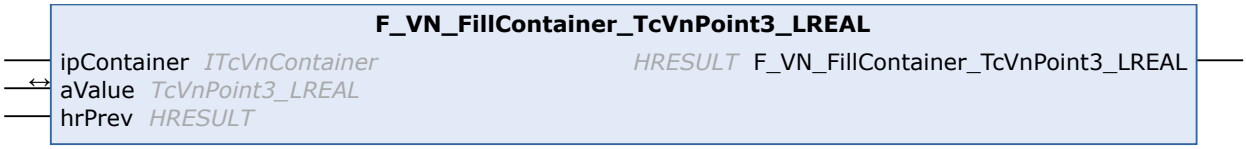
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.12 F_VN_FillContainer_TcVnPoint3_LREAL



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnPoint3_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnPoint3_LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint3_LREAL [▶ 171]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.13 F_VN_FillContainer_TcVnPoint3_REAL

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnPoint3_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnPoint3_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint3_REAL [▶ 171]	Value to set the container elements

Return value

HRESULT [▶ 150]

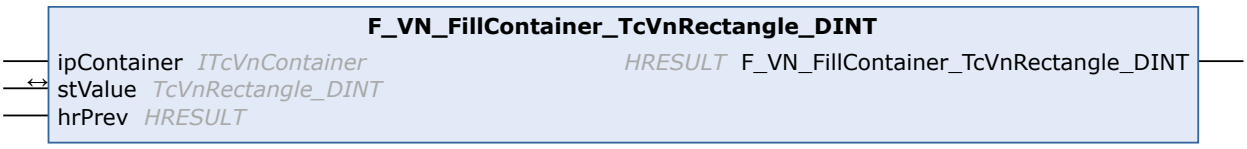
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.14 F_VN_FillContainer_TcVnRectangle_DINT



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnRectangle_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    stValue     : TcVnRectangle_DINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stValue	TcVnRectangle_DINT [▶ 257]	Value to set the container elements

Return value

HRESULT [▶ 150]

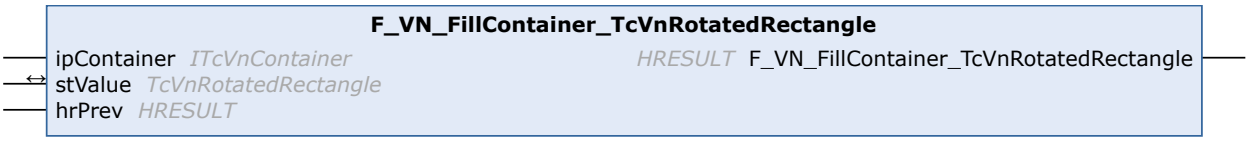
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.15 F_VN_FillContainer_TcVnRotatedRectangle



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnRotatedRectangle : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    stValue     : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stValue	TcVnRotatedRectangle [▶ 258]	Value to set the container elements

Return value

HRESULT [▶ 150]

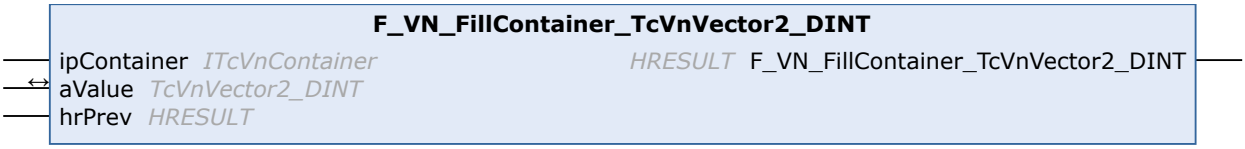
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.16 F_VN_FillContainer_TcVnVector2_DINT



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnVector2_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector2_DINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_DINT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.17 F_VN_FillContainer_TcVnVector2_INT

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnVector2_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector2_INT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_INT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

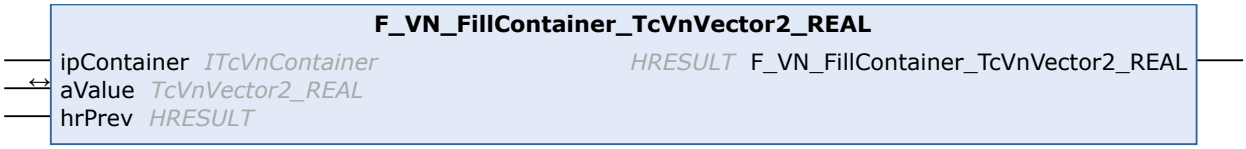
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.18 F_VN_FillContainer_TcVnVector2_REAL



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnVector2_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector2_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_REAL [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

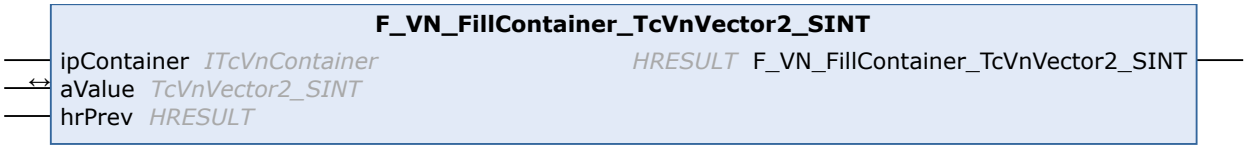
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.19 F_VN_FillContainer_TcVnVector2_SINT



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnVector2_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector2_SINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_SINT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

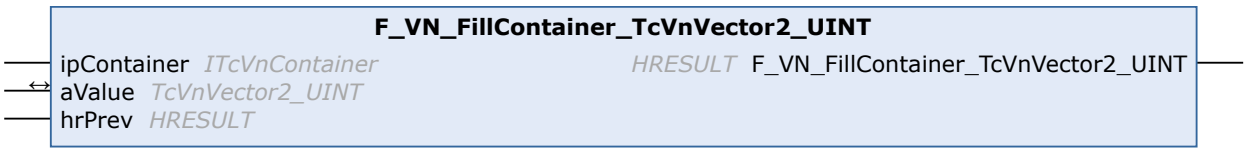
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.20 F_VN_FillContainer_TcVnVector2_UINT



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnVector2_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector2_UINT;
END_VAR
```

```
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_UINT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

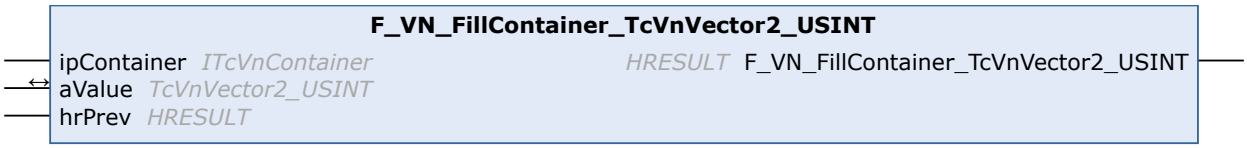
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.21 F_VN_FillContainer_TcVnVector2_USINT



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnVector2_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector2_USINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_USINT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

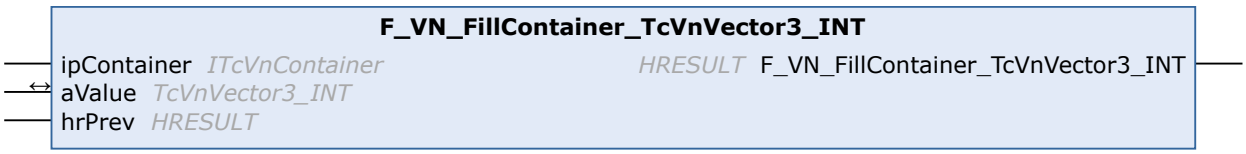
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.22 F_VN_FillContainer_TcVnVector3_INT



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnVector3_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector3_INT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_INT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

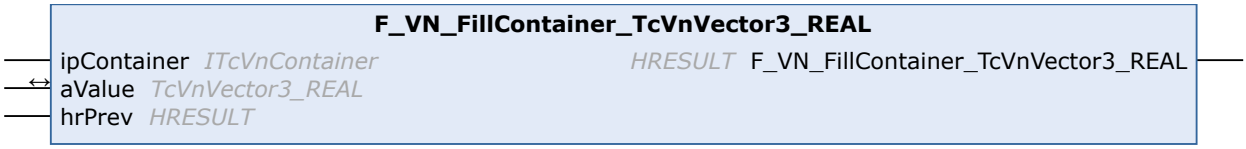
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.23 F_VN_FillContainer_TcVnVector3_REAL



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnVector3_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector3_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_REAL [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

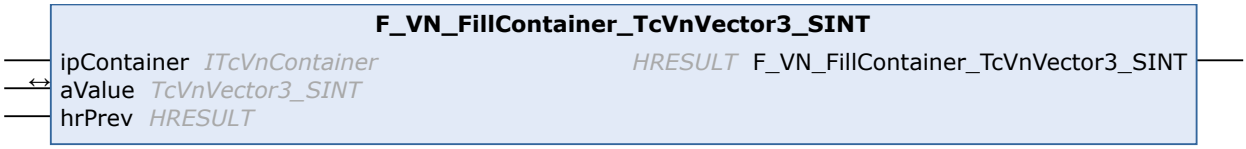
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.24 F_VN_FillContainer_TcVnVector3_SINT



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnVector3_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector3_SINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_SINT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

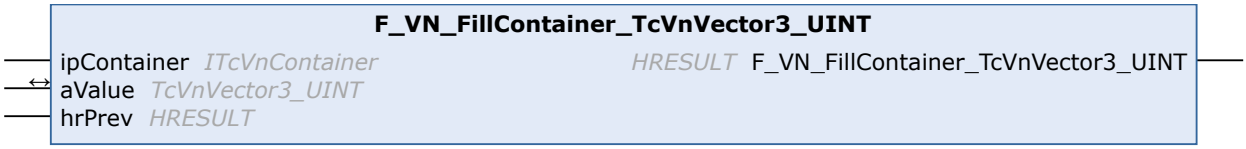
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.25 F_VN_FillContainer_TcVnVector3_UINT



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnVector3_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector3_UINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_UINT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

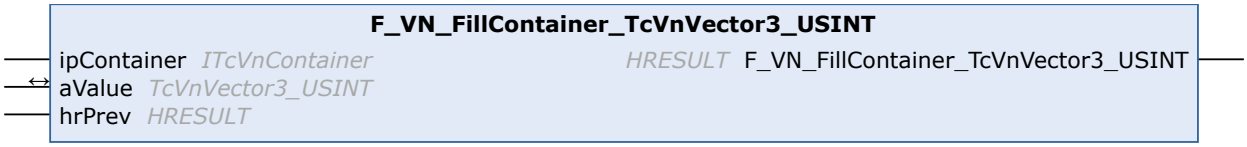
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.26 F_VN_FillContainer_TcVnVector3_USINT



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnVector3_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector3_USINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_USINT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

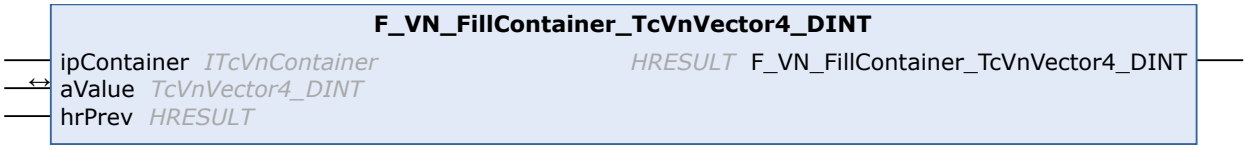
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.27 F_VN_FillContainer_TcVnVector4_DINT



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnVector4_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector4_DINT;
END_VAR
```

```
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_DINT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.28 F_VN_FillContainer_TcVnVector4_INT

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnVector4_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector4_INT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_INT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

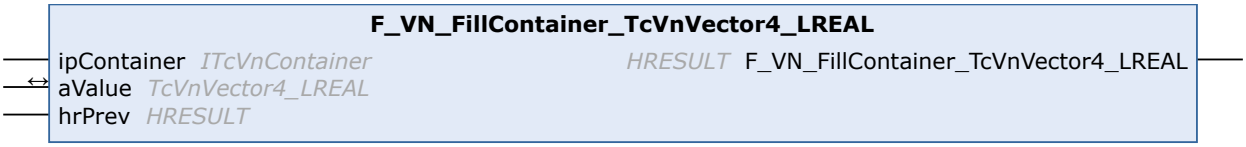
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.29 F_VN_FillContainer_TcVnVector4_LREAL



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnVector4_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_LREAL [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

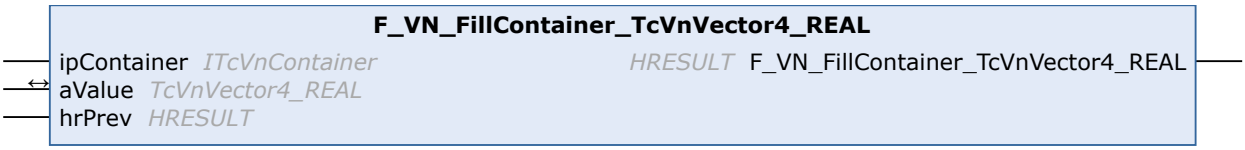
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.30 F_VN_FillContainer_TcVnVector4_REAL



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnVector4_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector4_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_REAL [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

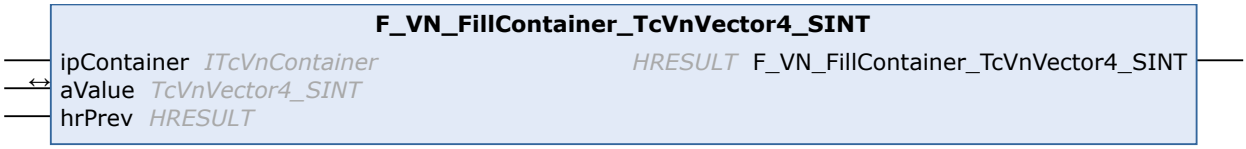
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.31 F_VN_FillContainer_TcVnVector4_SINT



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnVector4_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector4_SINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_SINT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

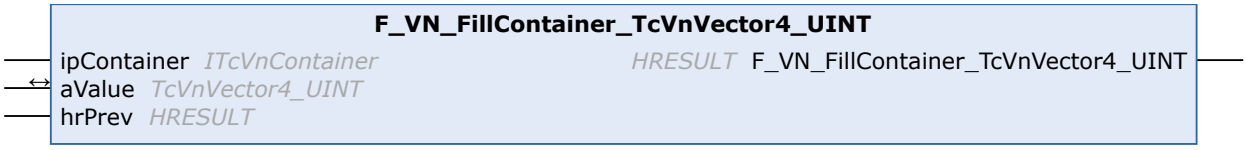
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.32 F_VN_FillContainer_TcVnVector4_UINT



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnVector4_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector4_UINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_UINT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

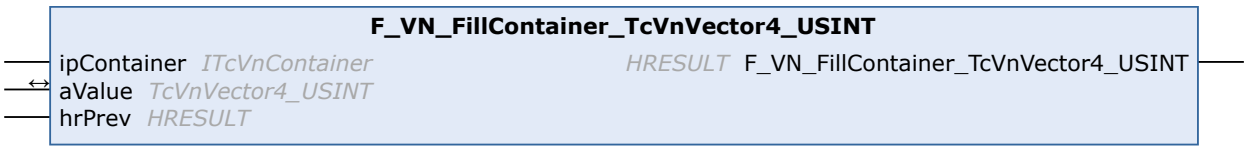
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.33 F_VN_FillContainer_TcVnVector4_USINT



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_TcVnVector4_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector4_USINT;
END_VAR
```

```
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_USINT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

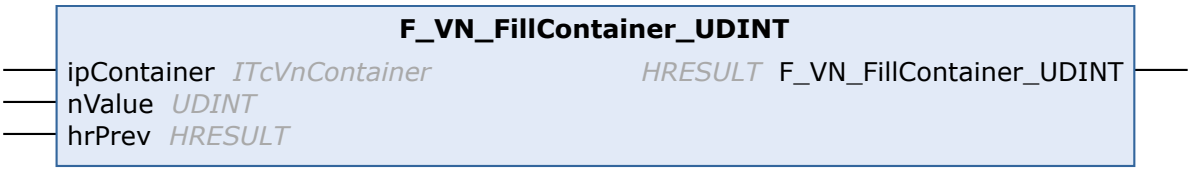
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.34 F_VN_FillContainer_UDINT



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_UDINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nValue      : UDINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nValue	UDINT	Value to set the container elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

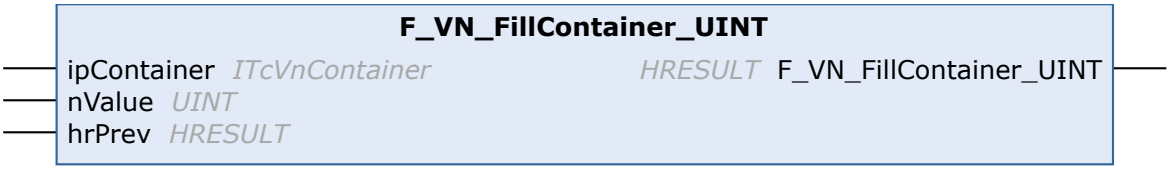
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.35 F_VN_FillContainer_UINT



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nValue      : UINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nValue	UINT	Value to set the container elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

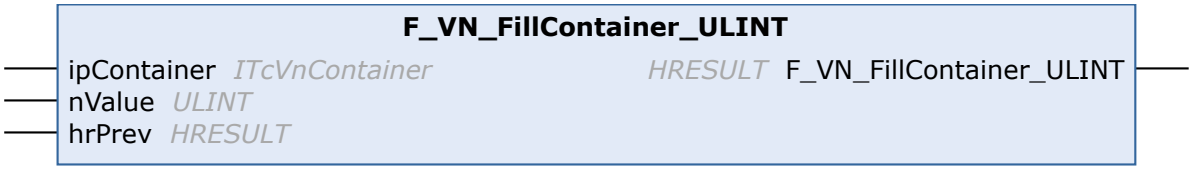
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.36 F_VN_FillContainer_ULINT



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_ULINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nValue      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nValue	ULINT	Value to set the container elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

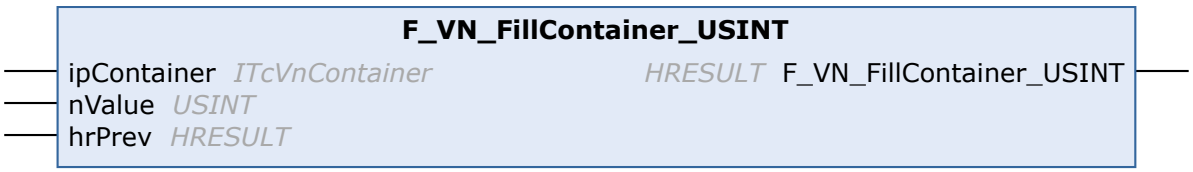
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.2.37 F_VN_FillContainer_USINT



Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_FillContainer_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nValue      : USINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nValue	USINT	Value to set the container elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3 F_VN_FillContainerExp

Dieses Kapitel enthält Experten-Funktionen, mit denen ein Container entsprechend dem Datentyp mit einem spezifischen Wert auch nur teilweise befüllt werden kann. Diese Funktion ist für Container mit Grundelementen verfügbar.

6.1.4.4.3.1 F_VN_FillContainerExp_DINT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nValue      : DINT;
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nValue	DINT	Value to set the container elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.2 F_VN_FillContainerExp_INT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nValue      : INT;
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nValue	INT	Value to set the container elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.3 F_VN_FillContainerExp_ITcVnImage

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_ITcVnImage : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    ipValue     : ITcVnImage;
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
ipValue	ITcVnImage [▶ 426]	Value to set the container elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.4 F_VN_FillContainerExp_LREAL

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    fValue      : LREAL;
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
fValue	LREAL	Value to set the container elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.5 F_VN_FillContainerExp_REAL

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    fValue      : REAL;
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
fValue	REAL	Value to set the container elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.6 F_VN_FillContainerExp_SINT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nValue      : SINT;
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nValue	SINT	Value to set the container elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.7 F_VN_FillContainerExp_TcVnDMatch

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnDMatch : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    stValue     : TcVnDMatch;
END_VAR
```

```
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stValue	TcVnDMatch [▶ 244]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.8 F_VN_FillContainerExp_TcVnKeyPoint

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnKeyPoint : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    stValue     : TcVnKeyPoint;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stValue	TcVnKeyPoint [▶ 244]	Value to set the container elements

Return value

HRESULT [▶ 150]

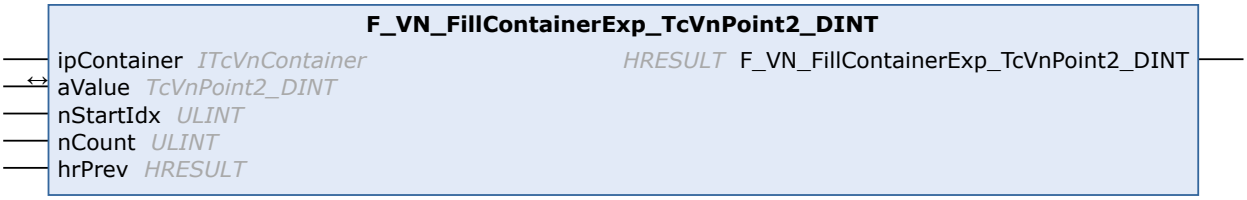
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.9 F_VN_FillContainerExp_TcVnPoint2_DINT



Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnPoint2_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnPoint2_DINT;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint2_DINT [▶ 171]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.10 F_VN_FillContainerExp_TcVnPoint2_LREAL

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnPoint2_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnPoint2_LREAL;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint2_LREAL [▶ 171]	Value to set the container elements

Return value

HRESULT [▶ 150]

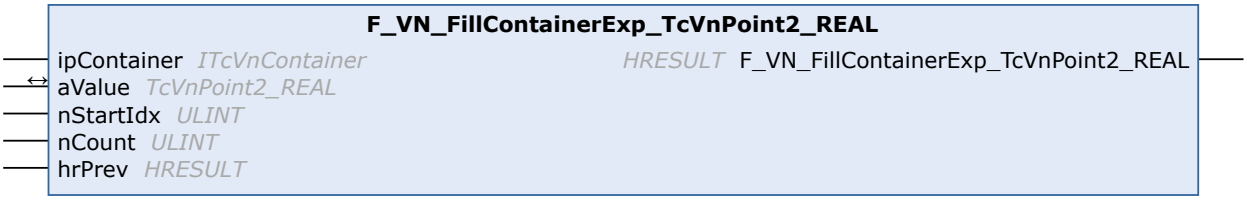
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.11 F_VN_FillContainerExp_TcVnPoint2_REAL



Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnPoint2_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint2_REAL [▶ 171]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.12 F_VN_FillContainerExp_TcVnPoint3_LREAL

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnPoint3_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnPoint3_LREAL;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint3_LREAL [▶ 171]	Value to set the container elements

Return value

HRESULT [▶ 150]

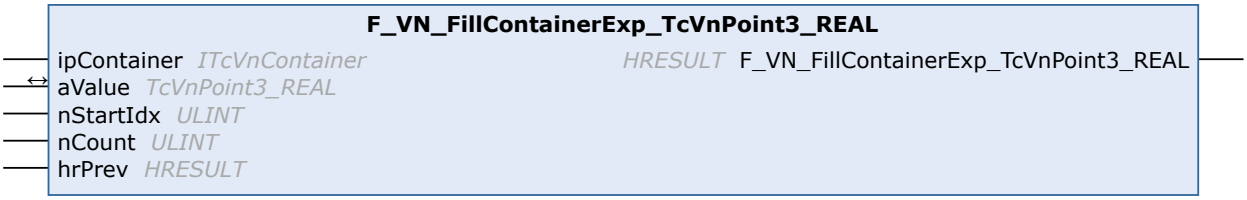
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.13 F_VN_FillContainerExp_TcVnPoint3_REAL



Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnPoint3_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnPoint3_REAL;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnPoint3_REAL [▶ 171]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.14 F_VN_FillContainerExp_TcVnRectangle_DINT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnRectangle_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    stValue     : TcVnRectangle_DINT;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stValue	TcVnRectangle_DINT [▶ 257]	Value to set the container elements

Return value

HRESULT [▶ 150]

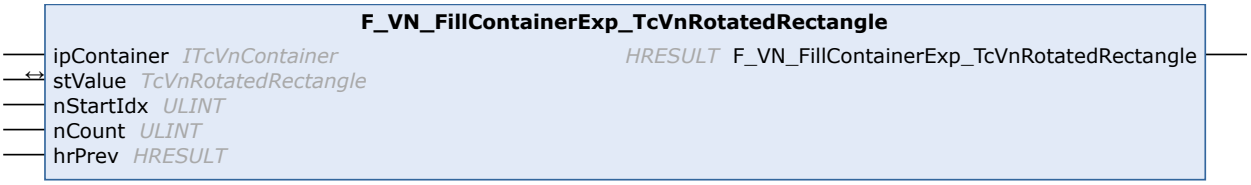
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.15 F_VN_FillContainerExp_TcVnRotatedRectangle



Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnRotatedRectangle : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    stValue     : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stValue	TcVnRotatedRectangle [▶ 258]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.16 F_VN_FillContainerExp_TcVnVector2_DINT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnVector2_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector2_DINT;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_DINT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

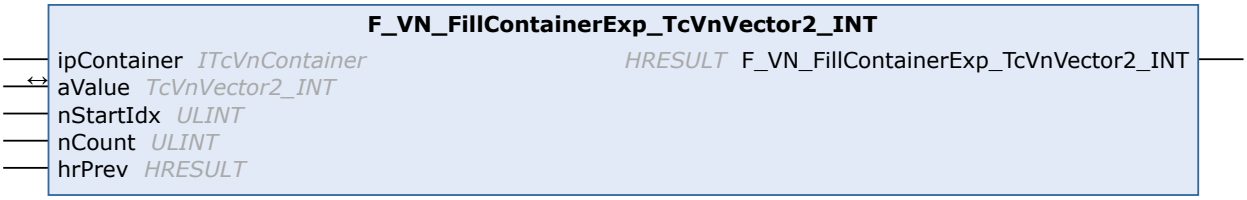
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.17 F_VN_FillContainerExp_TcVnVector2_INT



Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnVector2_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector2_INT;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_INT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.18 F_VN_FillContainerExp_TcVnVector2_REAL

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnVector2_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector2_REAL;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_REAL [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.19 F_VN_FillContainerExp_TcVnVector2_SINT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnVector2_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector2_SINT;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_SINT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.20 F_VN_FillContainerExp_TcVnVector2_UINT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnVector2_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector2_UINT;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_UINT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.21 F_VN_FillContainerExp_TcVnVector2_USINT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnVector2_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector2_USINT;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector2_USINT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

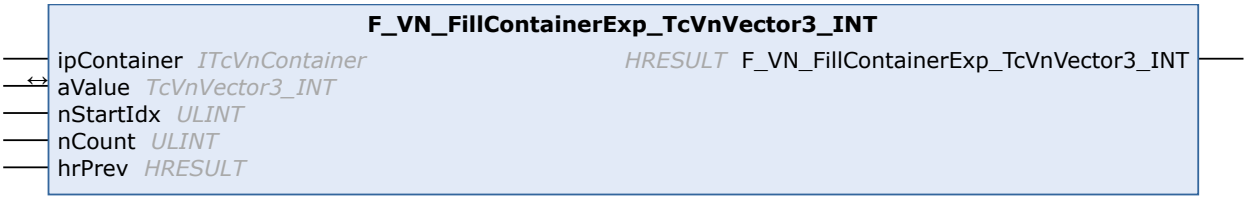
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.22 F_VN_FillContainerExp_TcVnVector3_INT



Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnVector3_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector3_INT;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_INT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.23 F_VN_FillContainerExp_TcVnVector3_REAL

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnVector3_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector3_REAL;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_REAL [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.24 F_VN_FillContainerExp_TcVnVector3_SINT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnVector3_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector3_SINT;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_SINT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.25 F_VN_FillContainerExp_TcVnVector3_UINT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnVector3_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector3_UINT;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_UINT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.26 F_VN_FillContainerExp_TcVnVector3_USINT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnVector3_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector3_USINT;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector3_USINT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.27 F_VN_FillContainerExp_TcVnVector4_DINT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnVector4_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector4_DINT;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_DINT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

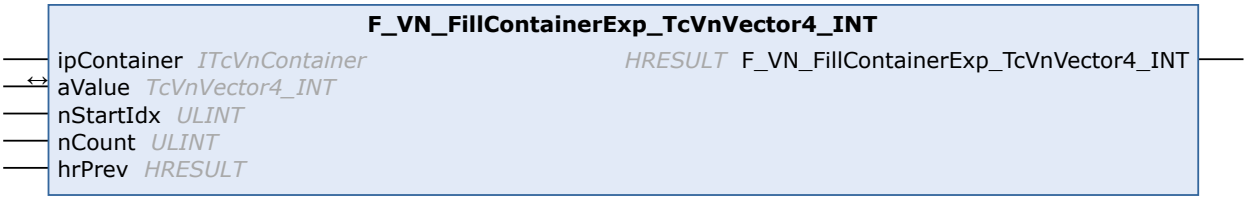
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.28 F_VN_FillContainerExp_TcVnVector4_INT



Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnVector4_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector4_INT;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_INT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

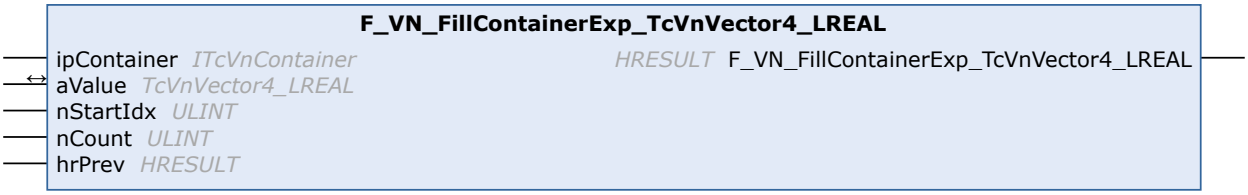
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.29 F_VN_FillContainerExp_TcVnVector4_LREAL



Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnVector4_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_LREAL [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.30 F_VN_FillContainerExp_TcVnVector4_REAL

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnVector4_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector4_REAL;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_REAL [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.31 F_VN_FillContainerExp_TcVnVector4_SINT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnVector4_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector4_SINT;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_SINT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.32 F_VN_FillContainerExp_TcVnVector4_UINT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnVector4_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector4_UINT;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_UINT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.33 F_VN_FillContainerExp_TcVnVector4_USINT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_TcVnVector4_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector4_USINT;
END_VAR
VAR_INPUT
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_USINT [▶ 173]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.34 F_VN_FillContainerExp_UDINT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_UDINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nValue      : UDINT;
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nValue	UDINT	Value to set the container elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.35 F_VN_FillContainerExp_UINT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nValue      : UINT;
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nValue	UINT	Value to set the container elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.36 F_VN_FillContainerExp_ULINT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_ULINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nValue      : ULINT;
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nValue	ULINT	Value to set the container elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.3.37 F_VN_FillContainerExp_USINT

Partially fill the container with the specified value. Only possible for containers with basic elements. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FillContainerExp_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nValue      : USINT;
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nValue	USINT	Value to set the container elements
nStartIdx	ULINT	Start index
nCount	ULINT	Number of elements to set
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

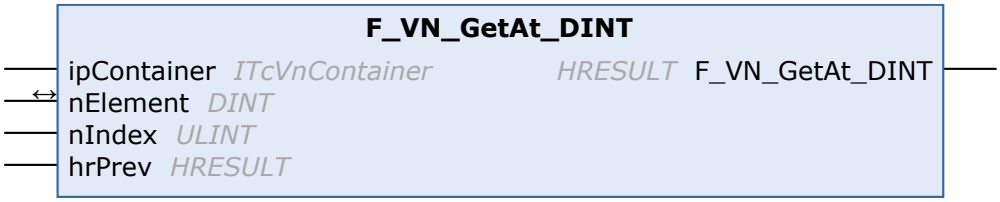
System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4 F_VN_GetAt

Die `F_VN_GetAt`-Funktionen sollten verwendet werden, wenn auf einzelne Elemente in einem Container zugegriffen werden muss. Falls alle Elemente eines Containers der Reihe nach verarbeitet werden sollen, bietet sich vielmehr die Verwendung eines Iterators [▶ 788] an.

6.1.4.4.4.1 F_VN_GetAt_DINT



Gets the DINT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nElement    : DINT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with DINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nElement	DINT	Returns the element at the specified index

Return value

HRESULT [▶ 150]

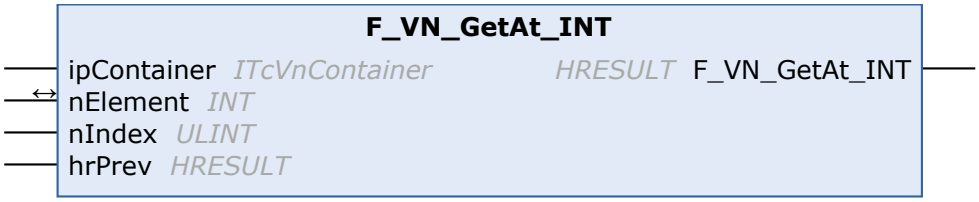
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.2 F_VN_GetAt_INT



Gets the INT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nElement    : INT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with INT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nElement	INT	Returns the element at the specified index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.3 F_VN_GetAt_ITcVnContainer

Gets the ITcVnContainer element at the specified index of the source container. (Alternatively use interface method .GetAt.)

Syntax

Definition:

```
FUNCTION F_VN_GetAt_ITcVnContainer : HRESULT
VAR_INPUT
    ipSrcContainer  : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    nIndex          : ULINT;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer	ITcVnContainer [▶ 385]	Container with ITcVnContainer elements
ipDestContainer	Reference To ITcVnContainer [▶ 385]	Returns the container at the specified index
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.4 F_VN_GetAt_ITcVnImage

Gets the ITcVnImage element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_ITcVnImage : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    ipElement   : Reference To ITcVnImage;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with ITcVnImage elements
ipElement	Reference To ITcVnImage [▶ 426]	Returns the element at the specified index
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

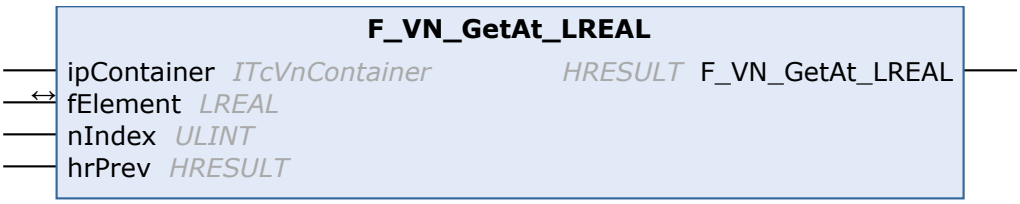
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.5 F_VN_GetAt_LREAL



Gets the LREAL element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    fElement    : LREAL;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with LREAL elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fElement	LREAL	Returns the element at the specified index

Return value

HRESULT [▶ 150]

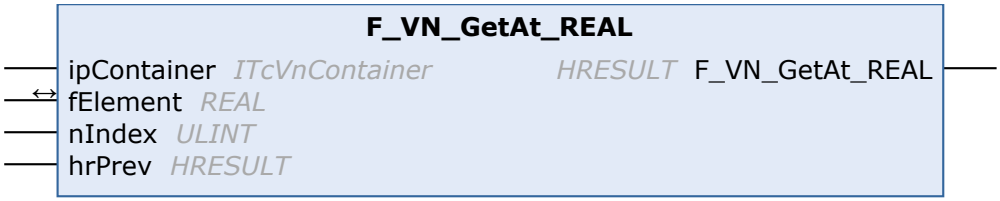
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.6 F_VN_GetAt_REAL



Gets the REAL element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    fElement    : REAL;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with REAL elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fElement	REAL	Returns the element at the specified index

Return value

HRESULT [▶ 150]

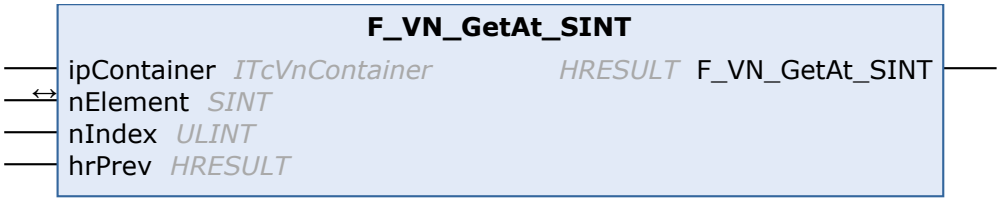
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.7 F_VN_GetAt_SINT



Gets the SINT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nElement    : SINT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with SINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nElement	SINT	Returns the element at the specified index

Return value

HRESULT [▶ 150]

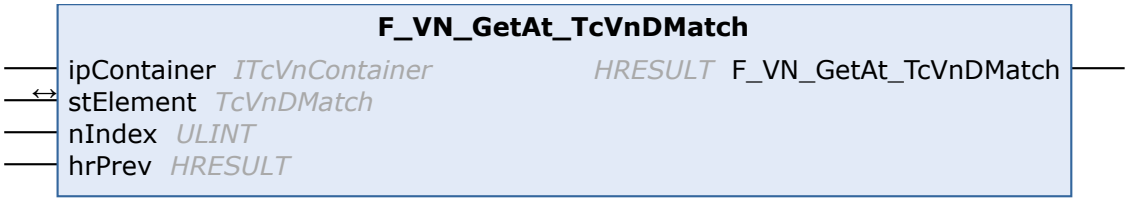
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.8 F_VN_GetAt_TcVnDMatch



Gets the TcVnDMatch element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnDMatch : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    stElement   : TcVnDMatch;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnDMatch elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnDMatch [▶ 244]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.9 F_VN_GetAt_TcVnKeyPoint

Gets the TcVnKeyPoint element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnKeyPoint : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    stElement   : TcVnKeyPoint;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnKeyPoint elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnKeyPoint [▶ 244]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.10 F_VN_GetAt_TcVnPoint2_DINT

Gets the TcVnPoint2_DINT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnPoint2_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnPoint2_DINT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnPoint2_DINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_DINT [▶ 171]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

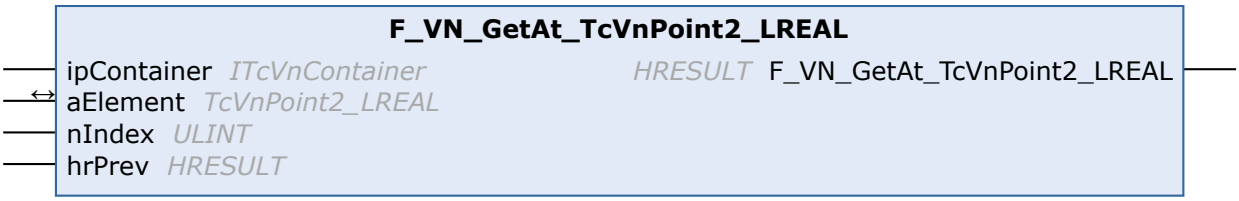
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.11 F_VN_GetAt_TcVnPoint2_LREAL



Gets the TcVnPoint2_LREAL element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnPoint2_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnPoint2_LREAL;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnPoint2_LREAL elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_LREAL [▶ 171]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.12 F_VN_GetAt_TcVnPoint2_REAL

Gets the TcVnPoint2_REAL element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnPoint2_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnPoint2_REAL elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_REAL [▶ 171]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

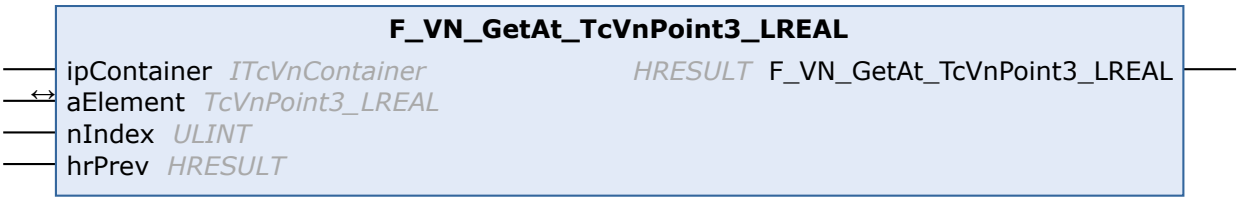
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.13 F_VN_GetAt_TcVnPoint3_LREAL



Gets the TcVnPoint3_LREAL element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnPoint3_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnPoint3_LREAL;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnPoint3_LREAL elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint3_LREAL [▶ 171]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.14 F_VN_GetAt_TcVnPoint3_REAL

Gets the TcVnPoint3_REAL element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnPoint3_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnPoint3_REAL;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnPoint3_REAL elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint3_REAL [▶ 171]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

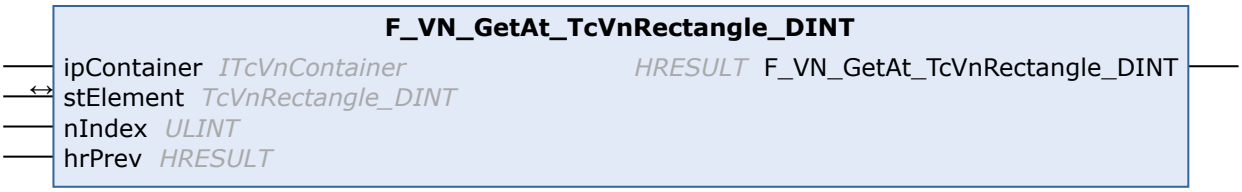
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.15 F_VN_GetAt_TcVnRectangle_DINT



Gets the TcVnRectangle_DINT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnRectangle_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    stElement   : TcVnRectangle_DINT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnRectangle_DINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnRectangle_DINT [▶ 257]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

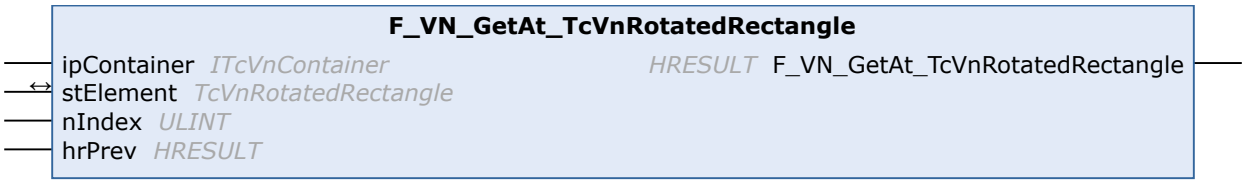
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.16 F_VN_GetAt_TcVnRotatedRectangle



Gets the TcVnRotatedRectangle element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnRotatedRectangle : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    stElement   : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnRotatedRectangle elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnRotatedRectangle [▶ 258]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.17 F_VN_GetAt_TcVnVector2_DINT

Gets the TcVnVector2_DINT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnVector2_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnVector2_DINT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector2_DINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_DINT [▶ 173]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.18 F_VN_GetAt_TcVnVector2_INT

Gets the TcVnVector2_INT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnVector2_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnVector2_INT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector2_INT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_INT [▶ 173]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

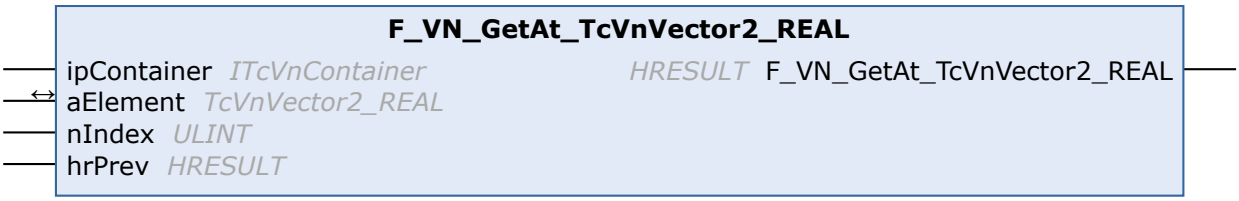
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.19 F_VN_GetAt_TcVnVector2_REAL



Gets the TcVnVector2_REAL element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnVector2_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnVector2_REAL;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector2_REAL elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_REAL [▶ 173]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

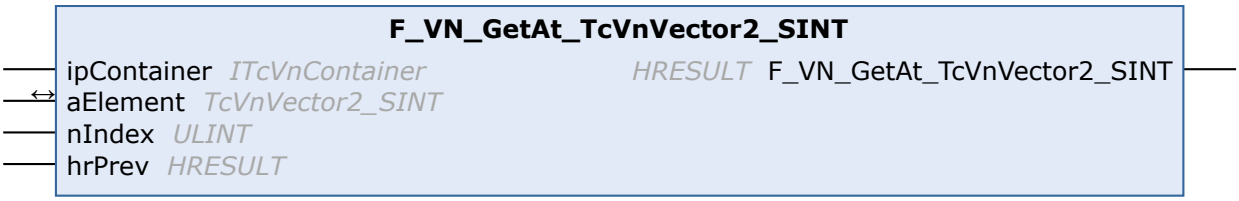
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.20 F_VN_GetAt_TcVnVector2_SINT



Gets the TcVnVector2_SINT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnVector2_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnVector2_SINT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector2_SINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_SINT [▶ 173]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

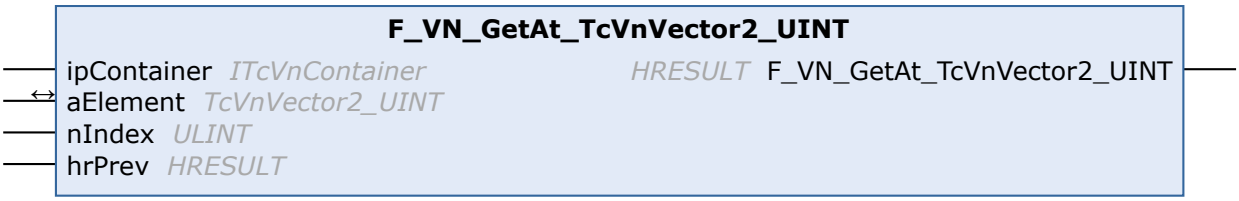
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.21 F_VN_GetAt_TcVnVector2_UINT



Gets the TcVnVector2_UINT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnVector2_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnVector2_UINT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector2_UINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_UINT [▶ 173]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

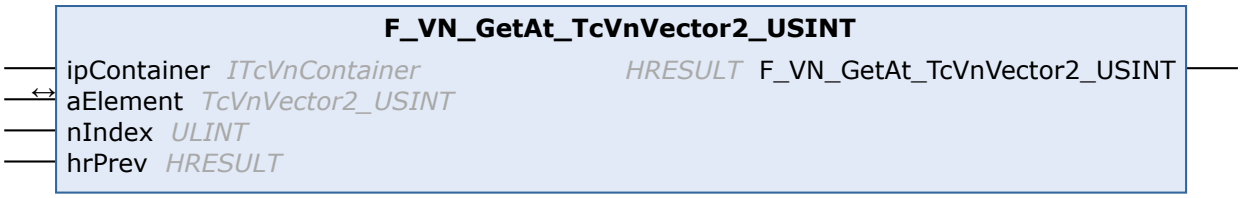
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.22 F_VN_GetAt_TcVnVector2_USINT



Gets the TcVnVector2_USINT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnVector2_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnVector2_USINT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector2_USINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_USINT [▶ 173]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.23 F_VN_GetAt_TcVnVector3_INT

Gets the TcVnVector3_INT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnVector3_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnVector3_INT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector3_INT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_INT [▶ 173]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

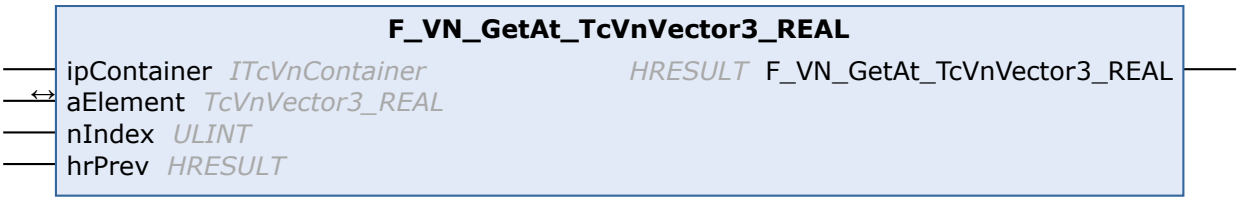
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.24 F_VN_GetAt_TcVnVector3_REAL



Gets the TcVnVector3_REAL element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnVector3_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnVector3_REAL;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector3_REAL elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_REAL [▶ 173]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

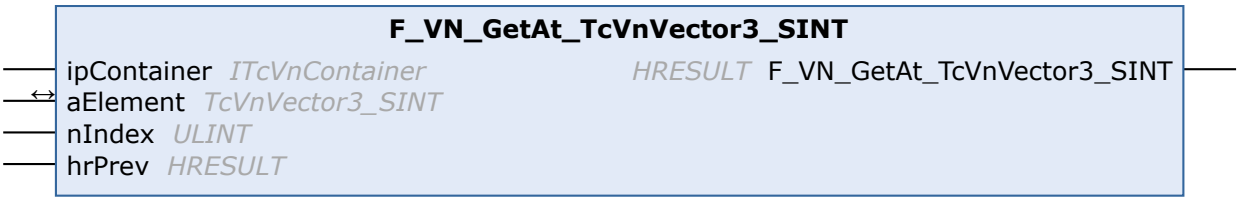
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.25 F_VN_GetAt_TcVnVector3_SINT



Gets the TcVnVector3_SINT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnVector3_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnVector3_SINT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector3_SINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_SINT [▶ 173]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

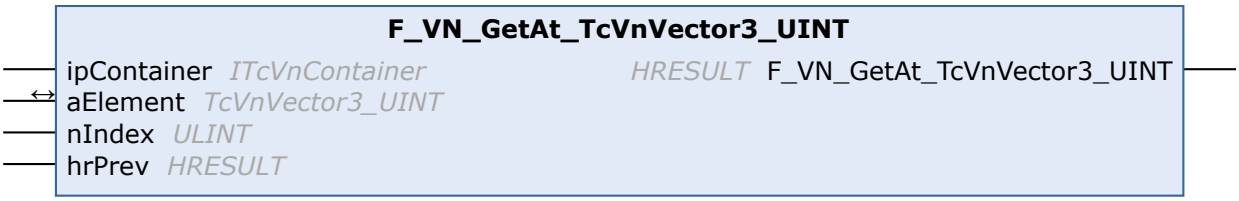
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.26 F_VN_GetAt_TcVnVector3_UINT



Gets the TcVnVector3_UINT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnVector3_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnVector3_UINT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector3_UINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_UINT [▶ 173]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

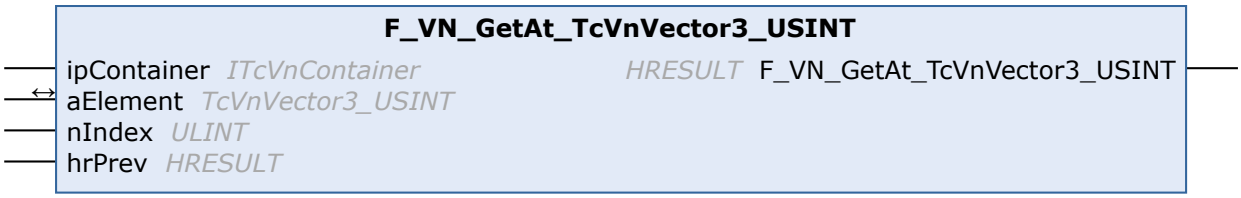
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.27 F_VN_GetAt_TcVnVector3_USINT



Gets the TcVnVector3_USINT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnVector3_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnVector3_USINT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector3_USINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_USINT [▶ 173]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.28 F_VN_GetAt_TcVnVector4_DINT

Gets the TcVnVector4_DINT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnVector4_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnVector4_DINT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector4_DINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_DINT [▶ 173]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.29 F_VN_GetAt_TcVnVector4_INT

Gets the TcVnVector4_INT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnVector4_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnVector4_INT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector4_INT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_INT [▶ 173]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

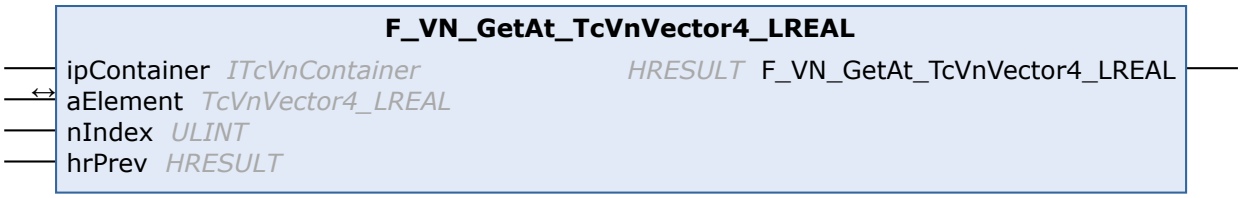
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.30 F_VN_GetAt_TcVnVector4_LREAL



Gets the TcVnVector4_LREAL element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnVector4_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector4_LREAL elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_LREAL [▶ 173]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.31 F_VN_GetAt_TcVnVector4_REAL

Gets the TcVnVector4_REAL element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnVector4_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnVector4_REAL;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector4_REAL elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_REAL [▶ 173]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

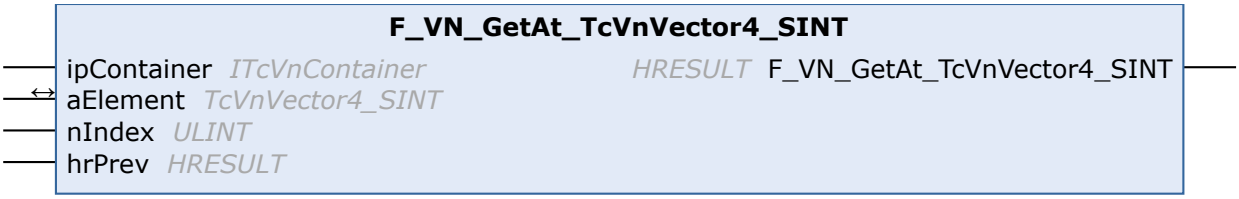
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.32 F_VN_GetAt_TcVnVector4_SINT



Gets the TcVnVector4_SINT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnVector4_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnVector4_SINT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector4_SINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_SINT [▶ 173]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

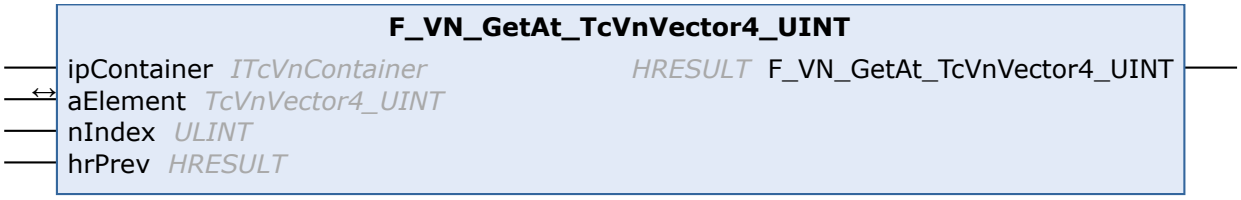
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.33 F_VN_GetAt_TcVnVector4_UINT



Gets the TcVnVector4_UINT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnVector4_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnVector4_UINT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector4_UINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_UINT [▶ 173]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

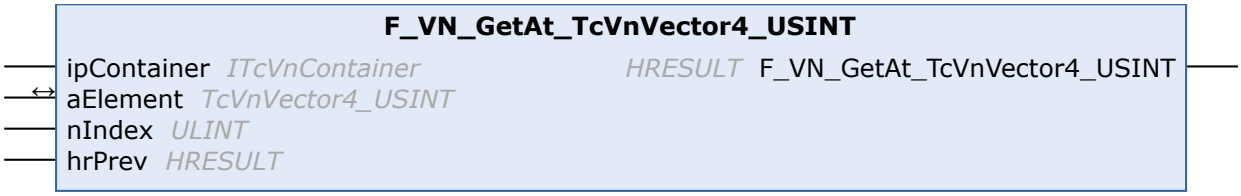
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.34 F_VN_GetAt_TcVnVector4_USINT



Gets the TcVnVector4_USINT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_TcVnVector4_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aElement    : TcVnVector4_USINT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector4_USINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_USINT [▶ 173]	Returns the element at the specified index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.35 F_VN_GetAt_UDINT

Gets the UDINT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_UDINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nElement    : UDINT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with UDINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nElement	UDINT	Returns the element at the specified index

Return value

HRESULT [▶ 150]

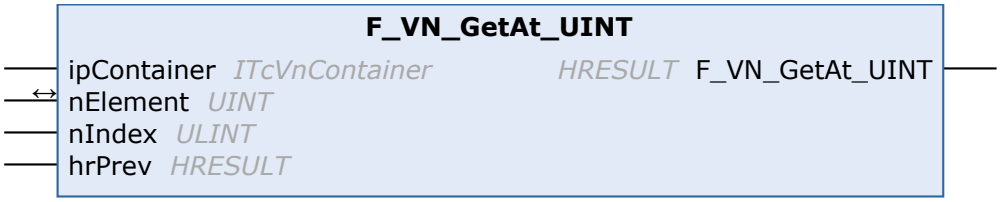
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.36 F_VN_GetAt_UINT



Gets the UINT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nElement    : UINT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with UINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nElement	UINT	Returns the element at the specified index

Return value

HRESULT [▶ 150]

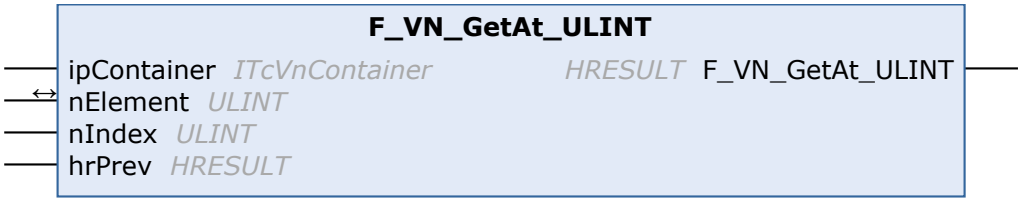
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.37 F_VN_GetAt_ULINT



Gets the ULINT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_ULINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nElement    : ULINT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with ULINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nElement	ULINT	Returns the element at the specified index

Return value

HRESULT [▶ 150]

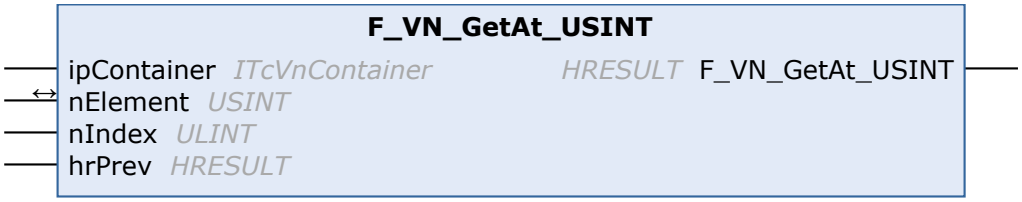
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.4.38 F_VN_GetAt_USINT



Gets the USINT element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_GetAt_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nElement    : USINT;
END_VAR
VAR_INPUT
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with USINT elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nElement	USINT	Returns the element at the specified index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

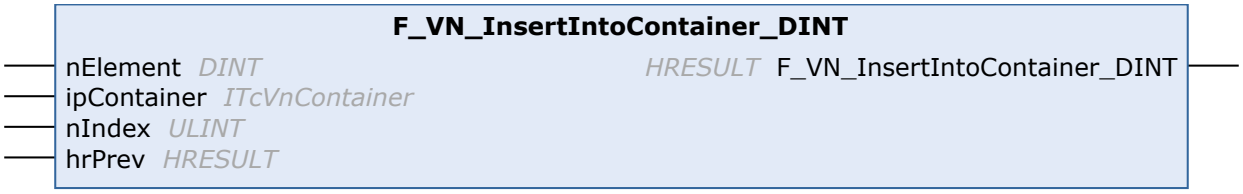
System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5 F_VN_InsertIntoContainer

Dieses Kapitel enthält Funktionen, mit denen ein Element an einer spezifischen Container-Position entsprechend dem Datentyp eingefügt werden kann.

6.1.4.4.5.1 F_VN_InsertIntoContainer_DINT



Insert an element of type DINT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_DINT : HRESULT
VAR_INPUT
    nElement    : DINT;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nElement	DINT	Single element to insert into ipContainer
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

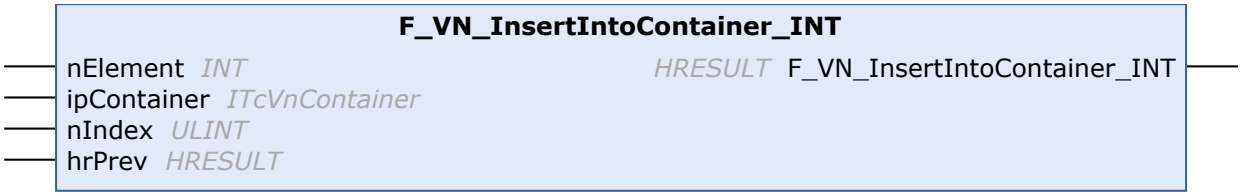
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.2 F_VN_InsertIntoContainer_INT



Insert an element of type INT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_INT : HRESULT
VAR_INPUT
    nElement    : INT;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nElement	INT	Single element to insert into ipContainer
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

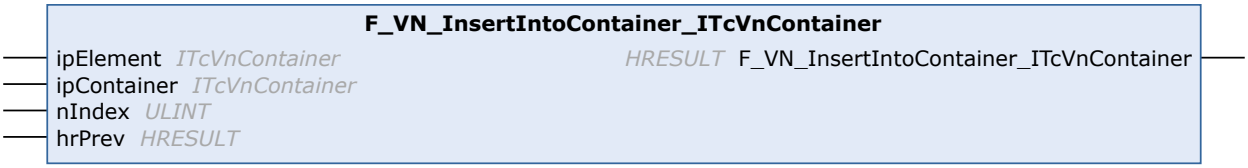
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.3 F_VN_InsertIntoContainer_ITcVnContainer



Insert an element (or a range of elements) into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_ITcVnContainer : HRESULT
VAR_INPUT
    ipElement   : ITcVnContainer;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipElement	ITcVnContainer [▶ 385]	Single element to insert into ipContainer or container with several elements
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

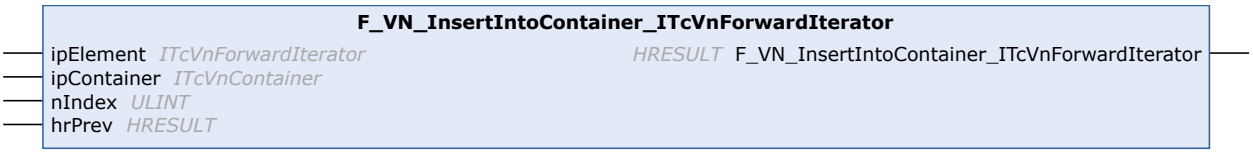
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.4 F_VN_InsertIntoContainer_ITcVnForwardIterator



Insert an element (represented by an iterator) into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_ITcVnForwardIterator : HRESULT
VAR_INPUT
    ipElement   : ITcVnForwardIterator;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipElement	ITcVnForwardIterator [▶ 376]	Single element to insert into ipContainer
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

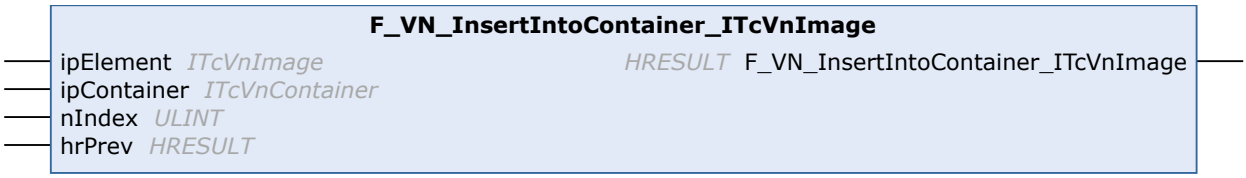
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.5 F_VN_InsertIntoContainer_ITcVnImage



Insert an element of type ITcVnImage into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_ITcVnImage : HRESULT
VAR_INPUT
    ipElement   : ITcVnImage;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipElement	ITcVnImage [▶ 426]	Single element to insert into ipContainer
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

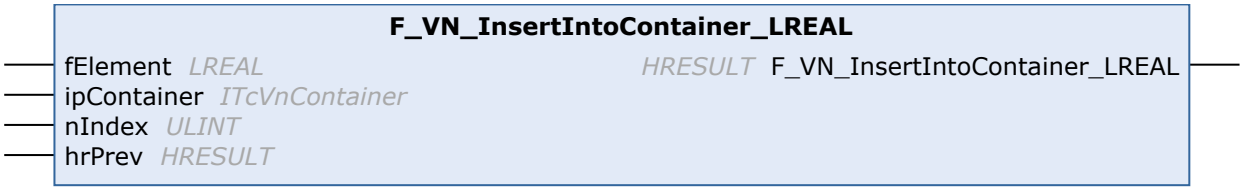
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.6 F_VN_InsertIntoContainer_LREAL



Insert an element of type LREAL into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_LREAL : HRESULT
VAR_INPUT
    fElement    : LREAL;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fElement	LREAL	Single element to insert into ipContainer
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

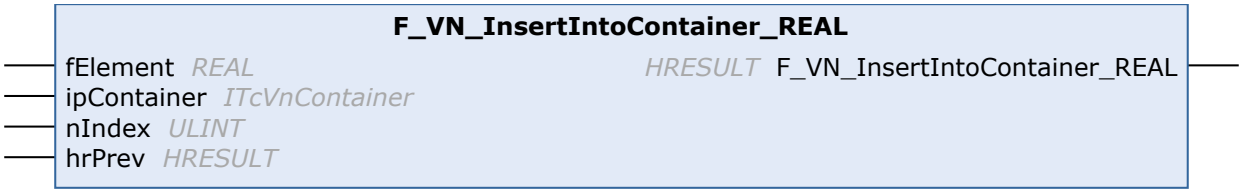
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.7 F_VN_InsertIntoContainer_REAL



Insert an element of type REAL into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_REAL : HRESULT
VAR_INPUT
    fElement    : REAL;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fElement	REAL	Single element to insert into ipContainer
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

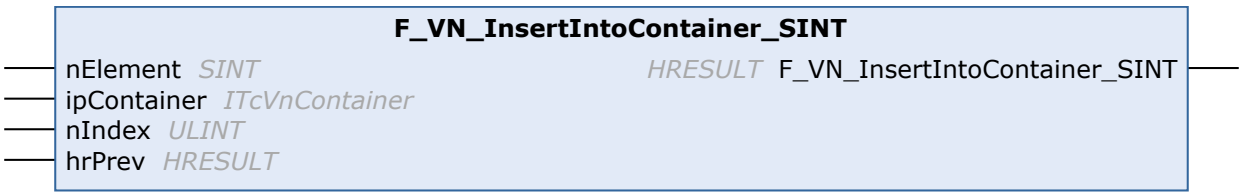
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.8 F_VN_InsertIntoContainer_SINT



Insert an element of type SINT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_SINT : HRESULT
VAR_INPUT
    nElement    : SINT;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nElement	SINT	Single element to insert into ipContainer
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

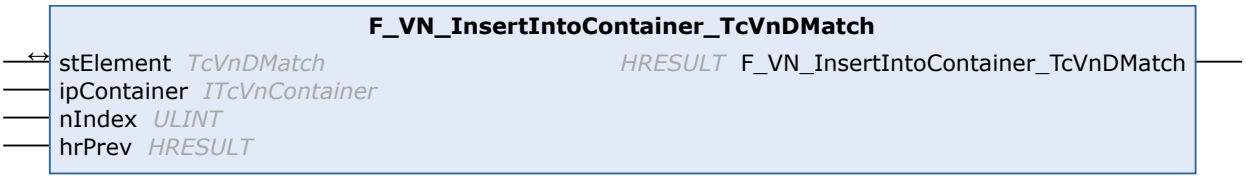
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.9 F_VN_InsertIntoContainer_TcVnDMatch



Insert an element of type TcVnDMatch into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnDMatch : HRESULT
VAR_IN_OUT
    stElement   : TcVnDMatch;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnDMatch [▶ 244]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

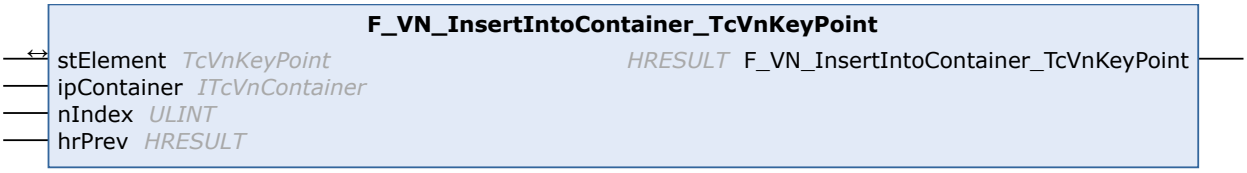
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.10 F_VN_InsertIntoContainer_TcVnKeyPoint



Insert an element of type TcVnKeyPoint into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnKeyPoint : HRESULT
VAR_IN_OUT
    stElement   : TcVnKeyPoint;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnKeyPoint [▶ 244]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

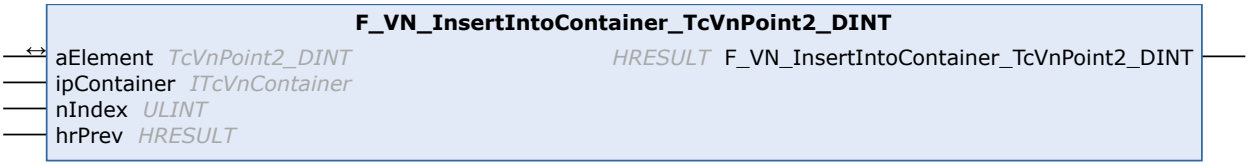
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.11 F_VN_InsertIntoContainer_TcVnPoint2_DINT



Insert an element of type TcVnPoint2_DINT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnPoint2_DINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnPoint2_DINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_DINT [▶ 171]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

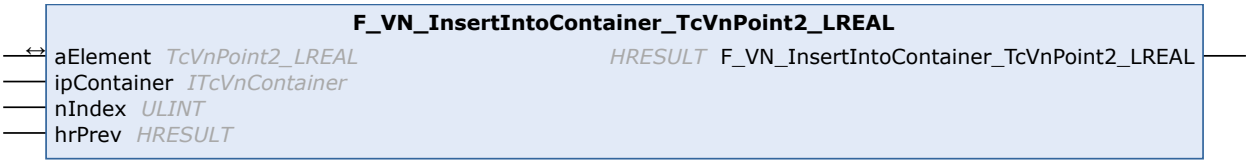
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.12 F_VN_InsertIntoContainer_TcVnPoint2_LREAL



Insert an element of type TcVnPoint2_LREAL into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnPoint2_LREAL : HRESULT
VAR_IN_OUT
    aElement    : TcVnPoint2_LREAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_LREAL [▶ 171]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

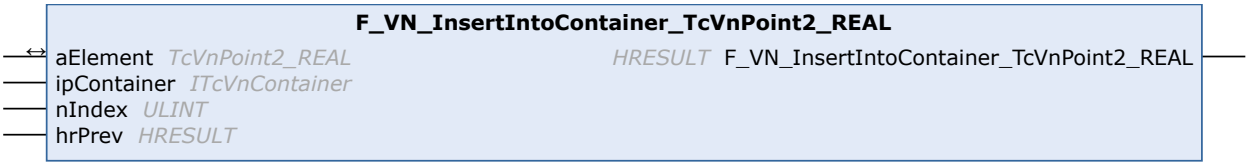
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.13 F_VN_InsertIntoContainer_TcVnPoint2_REAL



Insert an element of type TcVnPoint2_REAL into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnPoint2_REAL : HRESULT
VAR_IN_OUT
    aElement     : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_REAL [▶ 171]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

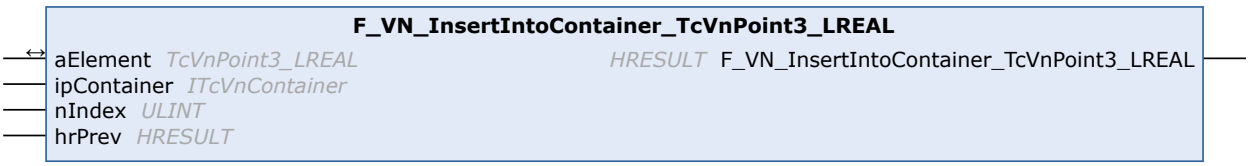
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.14 F_VN_InsertIntoContainer_TcVnPoint3_LREAL



Insert an element of type TcVnPoint3_LREAL into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnPoint3_LREAL : HRESULT
VAR_IN_OUT
    aElement    : TcVnPoint3_LREAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint3_LREAL [▶ 171]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

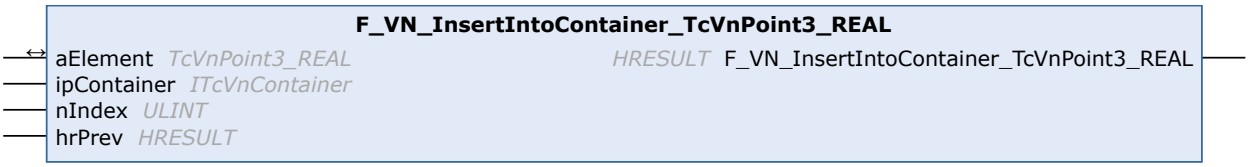
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.15 F_VN_InsertIntoContainer_TcVnPoint3_REAL



Insert an element of type TcVnPoint3_REAL into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnPoint3_REAL : HRESULT
VAR_IN_OUT
    aElement    : TcVnPoint3_REAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
```

```
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint3_REAL [▶ 171]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

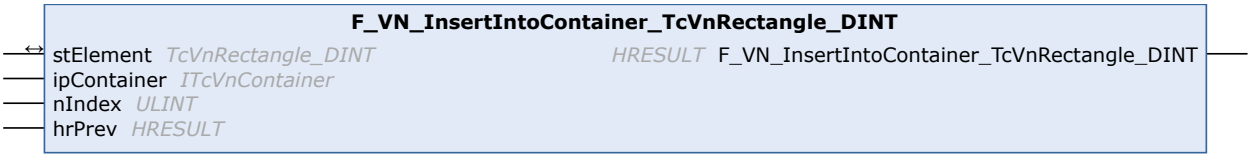
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.16 F_VN_InsertIntoContainer_TcVnRectangle_DINT



Insert an element of type TcVnRectangle_DINT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnRectangle_DINT : HRESULT
VAR_IN_OUT
    stElement   : TcVnRectangle_DINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnRectangle_DINT [▶ 257]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

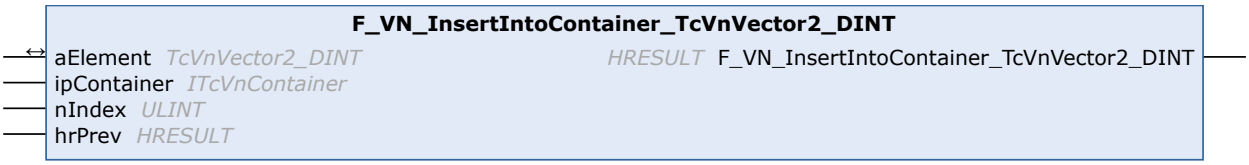
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.17 F_VN_InsertIntoContainer_TcVnVector2_DINT



Insert an element of type TcVnVector2_DINT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnVector2_DINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector2_DINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_DINT [▶ 173]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

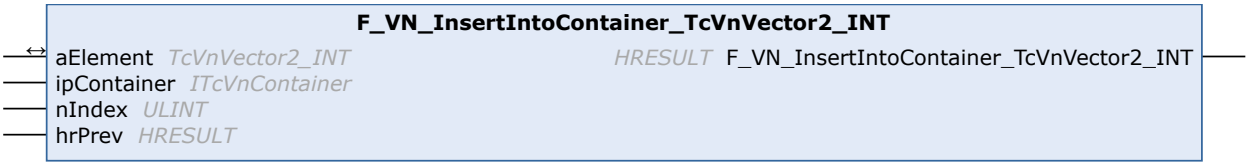
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.18 F_VN_InsertIntoContainer_TcVnVector2_INT



Insert an element of type TcVnVector2_INT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnVector2_INT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector2_INT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_INT [▶ 173]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

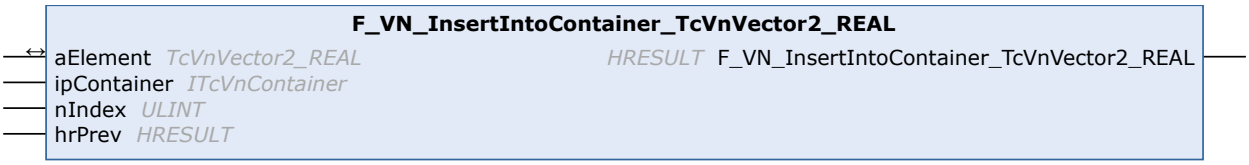
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.19 F_VN_InsertIntoContainer_TcVnVector2_REAL



Insert an element of type TcVnVector2_REAL into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnVector2_REAL : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector2_REAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_REAL [▶ 173]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

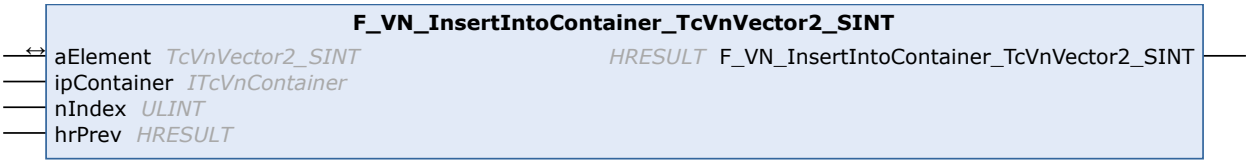
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.20 F_VN_InsertIntoContainer_TcVnVector2_SINT



Insert an element of type TcVnVector2_SINT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnVector2_SINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector2_SINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_SINT [▶ 173]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

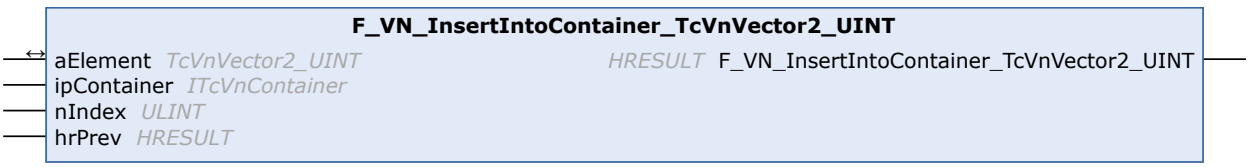
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.21 F_VN_InsertIntoContainer_TcVnVector2_UINT



Insert an element of type TcVnVector2_UINT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnVector2_UINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector2_UINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_UINT [▶ 173]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

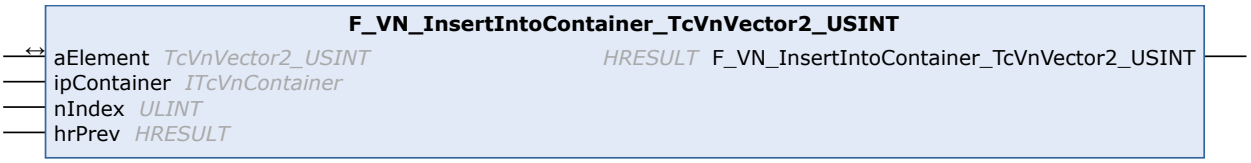
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.22 F_VN_InsertIntoContainer_TcVnVector2_USINT



Insert an element of type TcVnVector2_USINT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnVector2_USINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector2_USINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
```

```
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_USINT [▶ 173]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

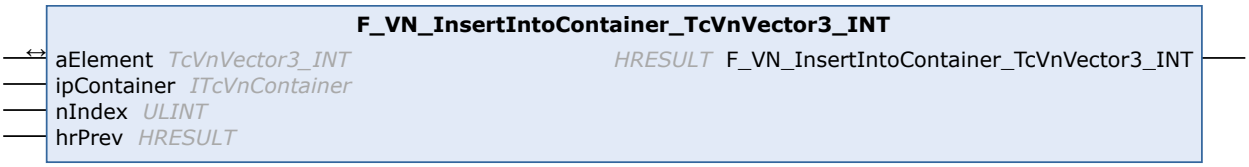
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.23 F_VN_InsertIntoContainer_TcVnVector3_INT



Insert an element of type TcVnVector3_INT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnVector3_INT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector3_INT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_INT [▶ 173]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

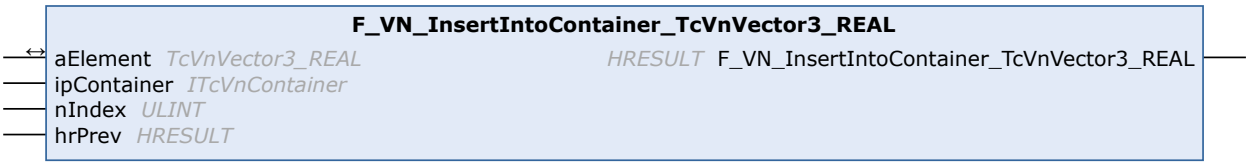
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.24 F_VN_InsertIntoContainer_TcVnVector3_REAL



Insert an element of type TcVnVector3_REAL into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnVector3_REAL : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector3_REAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_REAL [▶ 173]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

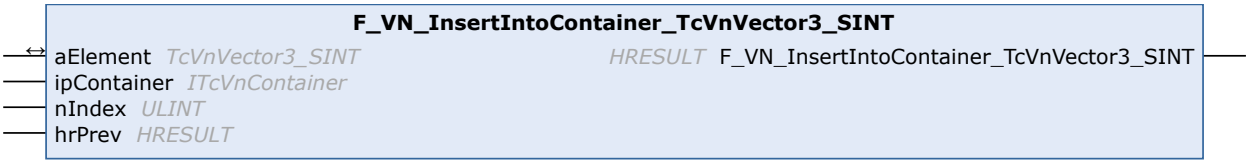
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.25 F_VN_InsertIntoContainer_TcVnVector3_SINT



Insert an element of type TcVnVector3_SINT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnVector3_SINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector3_SINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_SINT [▶ 173]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

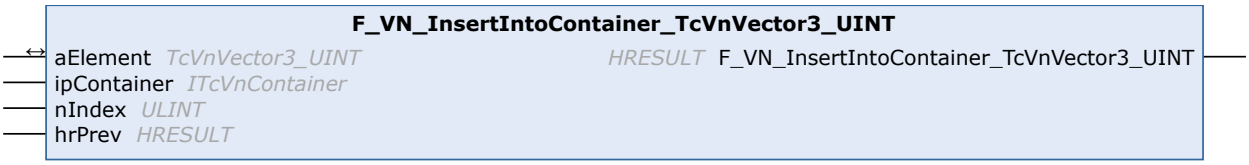
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.26 F_VN_InsertIntoContainer_TcVnVector3_UINT



Insert an element of type TcVnVector3_UINT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnVector3_UINT : HRESULT
VAR_IN_OUT
    aElement     : TcVnVector3_UINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_UINT [▶ 173]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

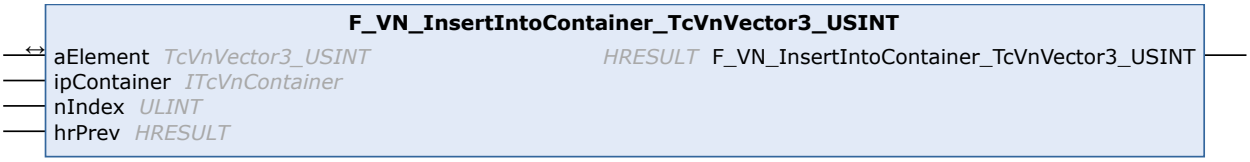
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.27 F_VN_InsertIntoContainer_TcVnVector3_USINT



Insert an element of type TcVnVector3_USINT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnVector3_USINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector3_USINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_USINT [▶ 173]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

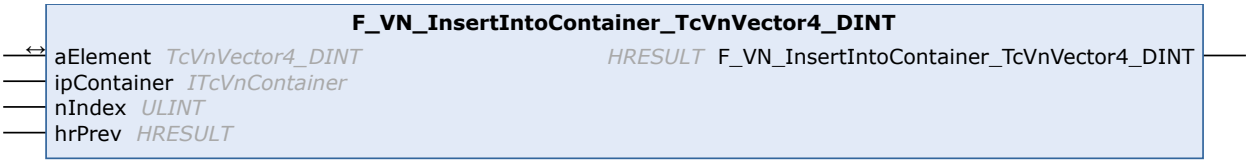
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.28 F_VN_InsertIntoContainer_TcVnVector4_DINT



Insert an element of type TcVnVector4_DINT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnVector4_DINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector4_DINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_DINT [▶ 173]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

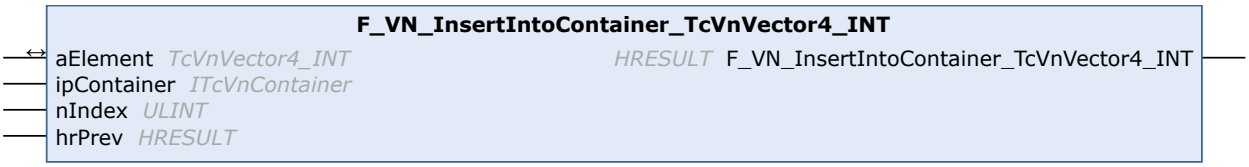
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.29 F_VN_InsertIntoContainer_TcVnVector4_INT



Insert an element of type TcVnVector4_INT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnVector4_INT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector4_INT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
```

```
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_INT [▶ 173]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

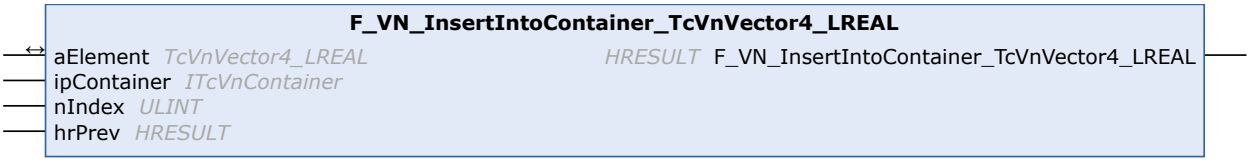
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.30 F_VN_InsertIntoContainer_TcVnVector4_LREAL



Insert an element of type TcVnVector4_LREAL into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnVector4_LREAL : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_LREAL [▶ 173]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

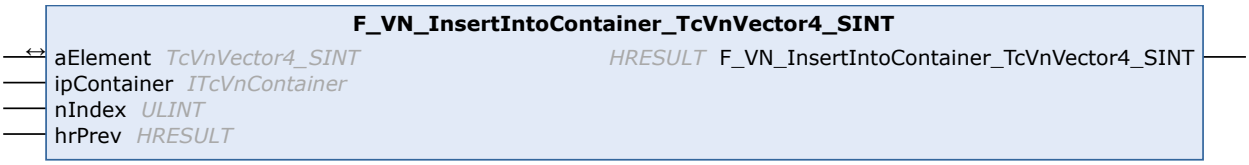
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.31 F_VN_InsertIntoContainer_TcVnVector4_SINT



Insert an element of type TcVnVector4_SINT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnVector4_SINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector4_SINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_SINT [▶ 173]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

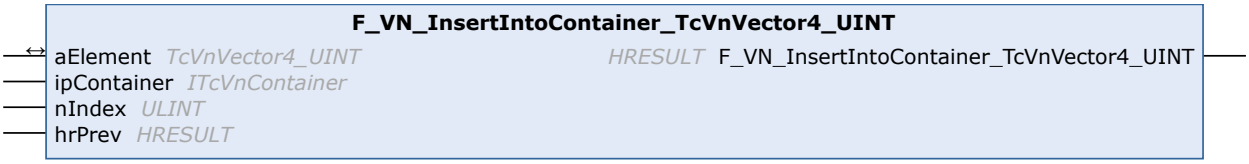
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.32 F_VN_InsertIntoContainer_TcVnVector4_UINT



Insert an element of type TcVnVector4_UINT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnVector4_UINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector4_UINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_UINT [▶ 173]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

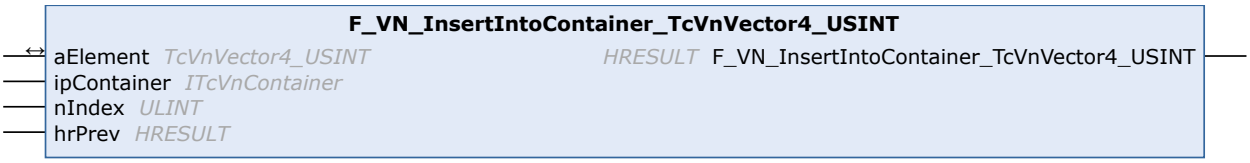
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.33 F_VN_InsertIntoContainer_TcVnVector4_USINT



Insert an element of type TcVnVector4_USINT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnVector4_USINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector4_USINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_USINT [▶ 173]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

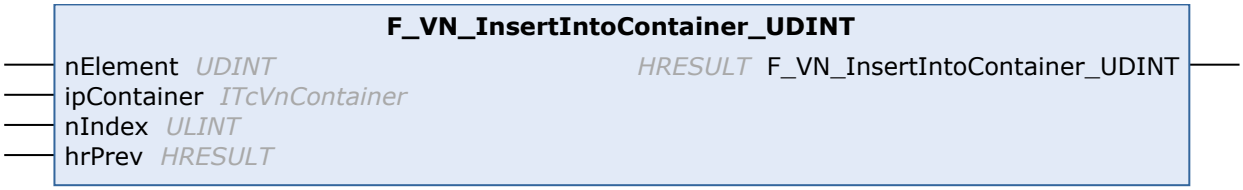
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.34 F_VN_InsertIntoContainer_UDINT



Insert an element of type UDINT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_UDINT : HRESULT
VAR_INPUT
    nElement    : UDINT;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nElement	UDINT	Single element to insert into ipContainer
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

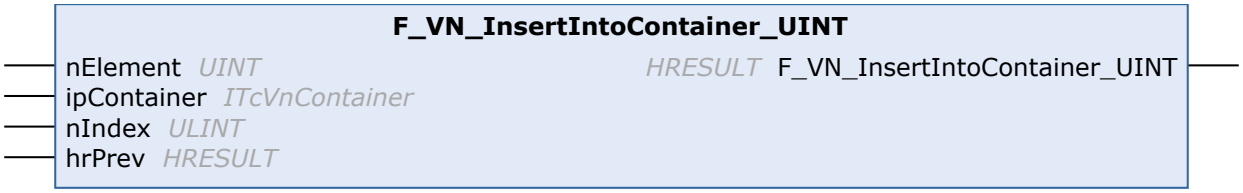
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.35 F_VN_InsertIntoContainer_UINT



Insert an element of type UINT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_UINT : HRESULT
VAR_INPUT
    nElement    : UINT;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nElement	UINT	Single element to insert into ipContainer
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

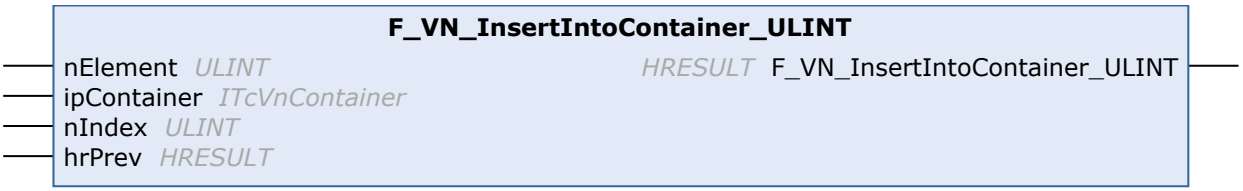
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.36 F_VN_InsertIntoContainer_ULINT



Insert an element of type ULINT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_ULINT : HRESULT
VAR_INPUT
    nElement    : ULINT;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nElement	ULINT	Single element to insert into ipContainer
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

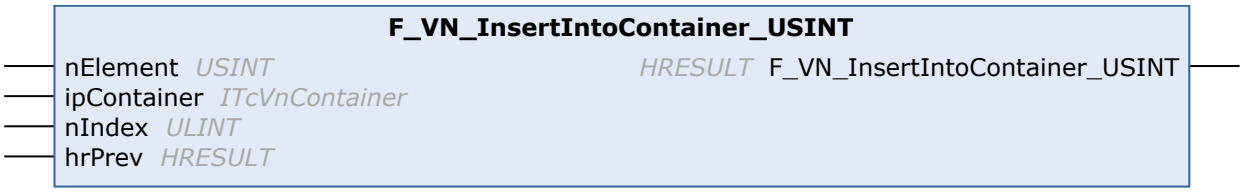
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.37 F_VN_InsertIntoContainer_USINT



Insert an element of type USINT into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_USINT : HRESULT
VAR_INPUT
    nElement    : USINT;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nElement	USINT	Single element to insert into ipContainer
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

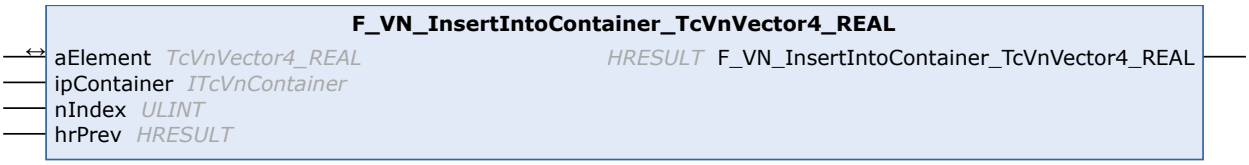
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.38 F_VN_InsertIntoContainer_TcVnVector4_REAL



Insert an element of type TcVnVector4_REAL into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnVector4_REAL : HRESULT
VAR_IN_OUT
    aElement     : TcVnVector4_REAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_REAL [▶ 173]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.5.39 F_VN_InsertIntoContainer_TcVnRotatedRectangle

Insert an element of type TcVnRotatedRectangle into a container before the specified position.

Syntax

Definition:

```
FUNCTION F_VN_InsertIntoContainer_TcVnRotatedRectangle : HRESULT
VAR_IN_OUT
    stElement   : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container in which to insert the element
nIndex	ULINT	Position, before which the element will be inserted
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnRotatedRectangle [▶ 258]	Single element to insert into ipContainer

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

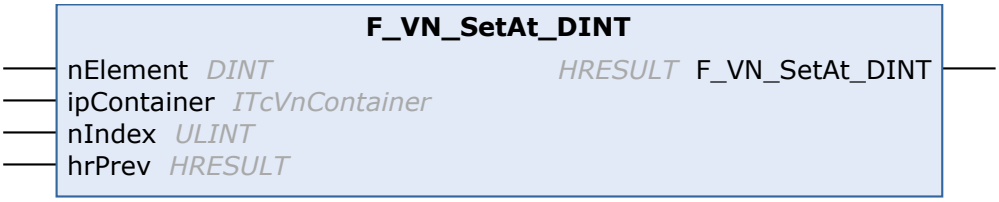
System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6 F_VN_SetAt

Die `F_VN_SetAt`-Funktionen sollten verwendet werden, wenn auf einzelne Elemente in einem Container zugegriffen werden muss. Falls alle Elemente eines Containers der Reihe nach bearbeitet werden sollen, bietet sich vielmehr die Verwendung eines Iterators [▶ 788] an.

6.1.4.4.6.1 F_VN_SetAt_DINT



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_DINT : HRESULT
VAR_INPUT
    nElement    : DINT;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nElement	DINT	Element to set at the specified container position
ipContainer	ITcVnContainer [▶ 385]	Container with DINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

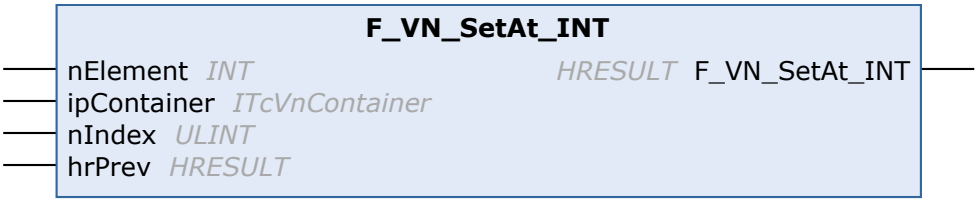
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.2 F_VN_SetAt_INT



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_INT : HRESULT
VAR_INPUT
    nElement    : INT;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nElement	INT	Element to set at the specified container position
ipContainer	ITcVnContainer [▶ 385]	Container with INT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

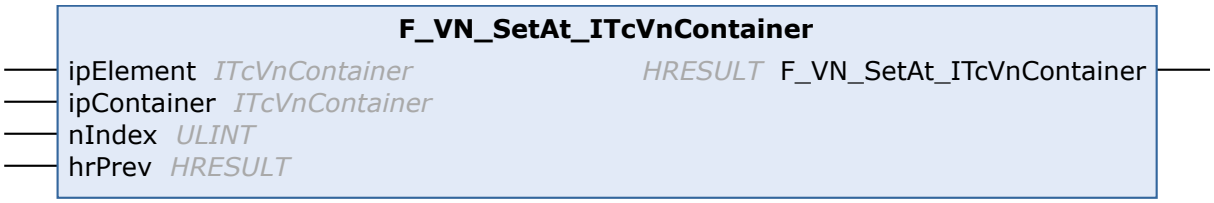
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.3 F_VN_SetAt_ITcVnContainer



Sets the element at the specified index of the container. (Alternatively use interface method .SetAt.)

Syntax

Definition:

```
FUNCTION F_VN_SetAt_ITcVnContainer : HRESULT
VAR_INPUT
    ipElement   : ITcVnContainer;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipElement	ITcVnContainer [▶ 385]	Element to set at the specified container position
ipContainer	ITcVnContainer [▶ 385]	Container with ITcVnContainer elements, in which the element at position nIndex is replaced by ipElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

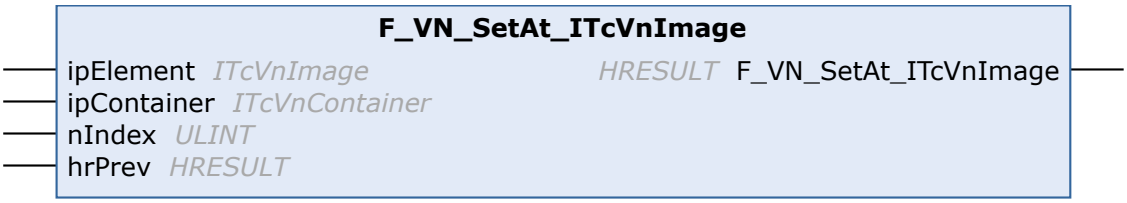
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.4 F_VN_SetAt_ITcVnImage



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_ITcVnImage : HRESULT
VAR_INPUT
    ipElement   : ITcVnImage;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipElement	ITcVnImage [▶ 426]	Element to set at the specified container position
ipContainer	ITcVnContainer [▶ 385]	Container with ITcVnImage elements, in which the element at position nIndex is replaced by ipElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

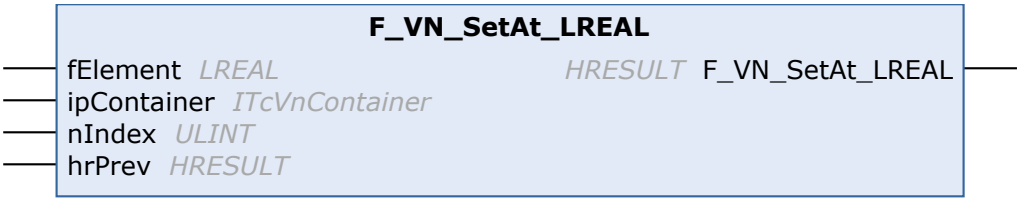
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.5 F_VN_SetAt_LREAL



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_LREAL : HRESULT
VAR_INPUT
    fElement    : LREAL;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fElement	LREAL	Element to set at the specified container position
ipContainer	ITcVnContainer [▶ 385]	Container with LREAL elements, in which the element at position nIndex is replaced by fElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

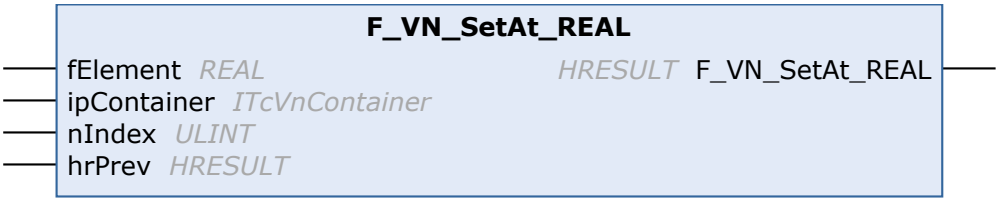
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.6 F_VN_SetAt_REAL



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_REAL : HRESULT
VAR_INPUT
    fElement    : REAL;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fElement	REAL	Element to set at the specified container position
ipContainer	ITcVnContainer [▶ 385]	Container with REAL elements, in which the element at position nIndex is replaced by fElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

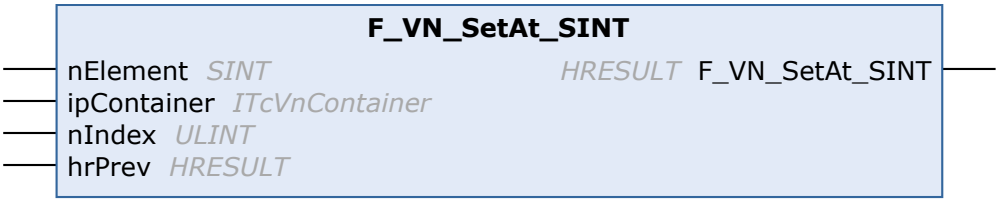
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.7 F_VN_SetAt_SINT



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_SINT : HRESULT
VAR_INPUT
    nElement    : SINT;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nElement	SINT	Element to set at the specified container position
ipContainer	ITcVnContainer [▶ 385]	Container with SINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

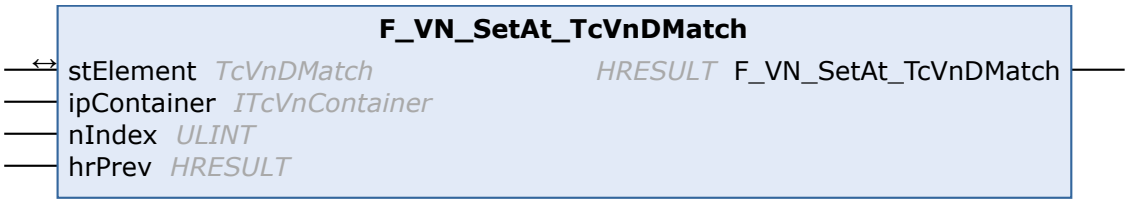
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.8 F_VN_SetAt_TcVnDMatch



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnDMatch : HRESULT
VAR_IN_OUT
    stElement   : TcVnDMatch;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnDMatch elements, in which the element at position nIndex is replaced by stElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnDMatch [▶ 244]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.9 F_VN_SetAt_TcVnKeyPoint

Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnKeyPoint : HRESULT
VAR_IN_OUT
    stElement   : TcVnKeyPoint;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnKeyPoint elements, in which the element at position nIndex is replaced by stElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnKeyPoint [▶ 244]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.10 F_VN_SetAt_TcVnPoint2_DINT

Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnPoint2_DINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnPoint2_DINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnPoint2_DINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_DINT [▶ 171]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

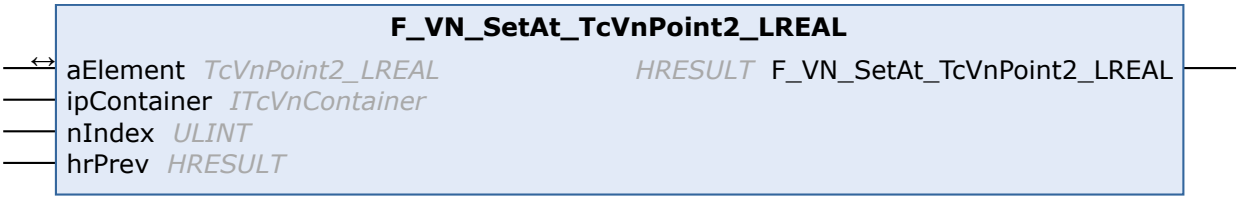
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.11 F_VN_SetAt_TcVnPoint2_LREAL



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnPoint2_LREAL : HRESULT
VAR_IN_OUT
    aElement    : TcVnPoint2_LREAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnPoint2_LREAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_LREAL [▶ 171]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.12 F_VN_SetAt_TcVnPoint2_REAL

Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnPoint2_REAL : HRESULT
VAR_IN_OUT
    aElement     : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnPoint2_REAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint2_REAL [▶ 171]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

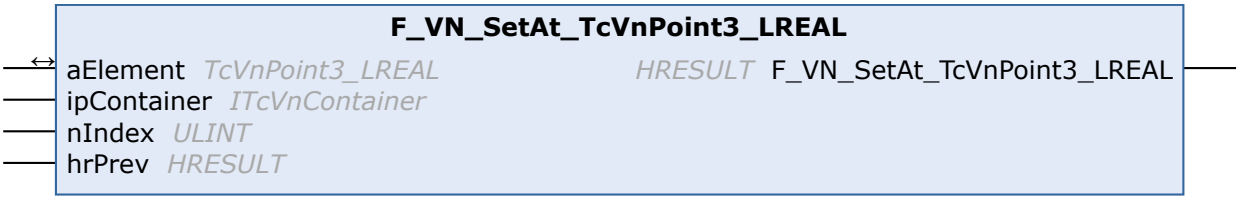
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.13 F_VN_SetAt_TcVnPoint3_LREAL



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnPoint3_LREAL : HRESULT
VAR_IN_OUT
    aElement    : TcVnPoint3_LREAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnPoint3_LREAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint3_LREAL [▶ 171]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.14 F_VN_SetAt_TcVnPoint3_REAL

Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnPoint3_REAL : HRESULT
VAR_IN_OUT
    aElement     : TcVnPoint3_REAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnPoint3_REAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnPoint3_REAL [▶ 171]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

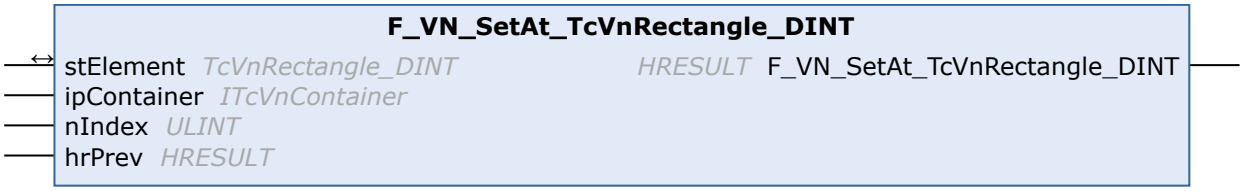
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.15 F_VN_SetAt_TcVnRectangle_DINT



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnRectangle_DINT : HRESULT
VAR_IN_OUT
    stElement   : TcVnRectangle_DINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnRectangle_DINT elements, in which the element at position nIndex is replaced by stElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnRectangle_DINT [▶ 257]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

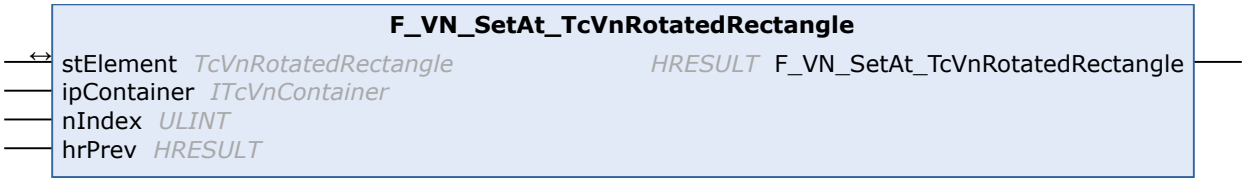
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.16 F_VN_SetAt_TcVnRotatedRectangle



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnRotatedRectangle : HRESULT
VAR_IN_OUT
    stElement   : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnRotatedRectangle elements, in which the element at position nIndex is replaced by stElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stElement	TcVnRotatedRectangle [▶ 258]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

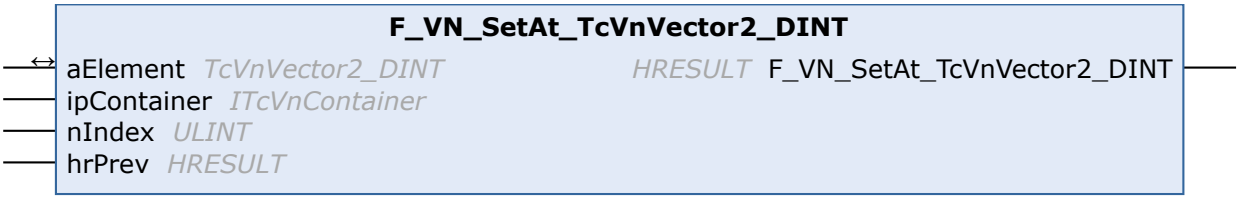
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.17 F_VN_SetAt_TcVnVector2_DINT



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnVector2_DINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector2_DINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector2_DINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_DINT [▶ 173]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

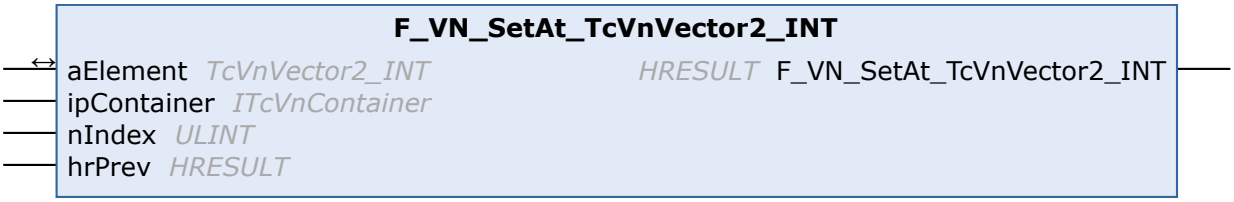
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.18 F_VN_SetAt_TcVnVector2_INT



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnVector2_INT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector2_INT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector2_INT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_INT [▶ 173]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

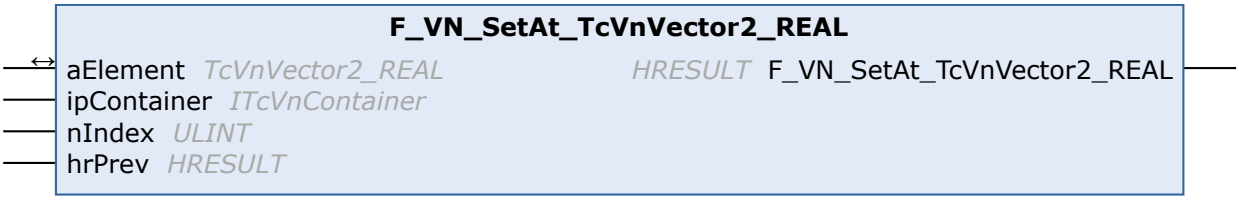
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.19 F_VN_SetAt_TcVnVector2_REAL



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnVector2_REAL : HRESULT
VAR_IN_OUT
    aElement     : TcVnVector2_REAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector2_REAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_REAL [▶ 173]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

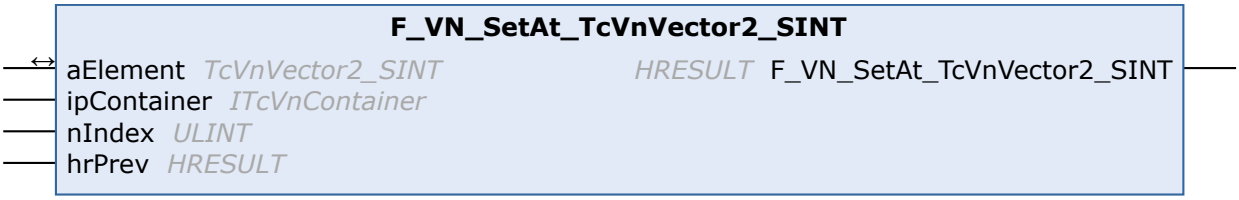
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.20 F_VN_SetAt_TcVnVector2_SINT



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnVector2_SINT : HRESULT
VAR_IN_OUT
    aElement     : TcVnVector2_SINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector2_SINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_SINT [▶ 173]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

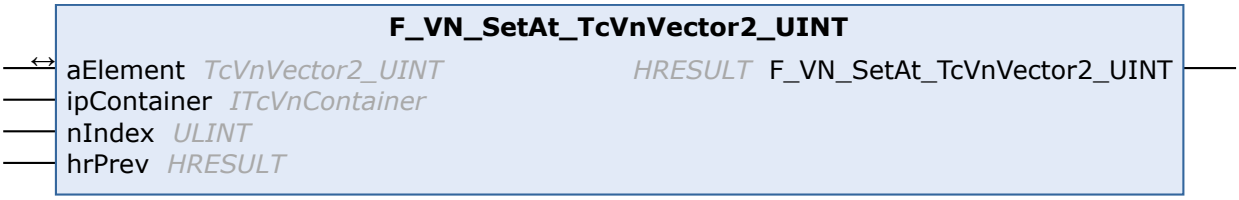
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.21 F_VN_SetAt_TcVnVector2_UINT



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnVector2_UINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector2_UINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector2_UINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_UINT [▶ 173]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

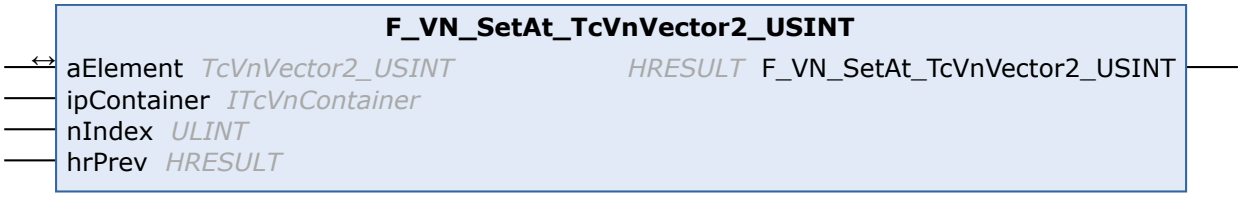
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.22 F_VN_SetAt_TcVnVector2_USINT



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnVector2_USINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector2_USINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector2_USINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector2_USINT [▶ 173]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

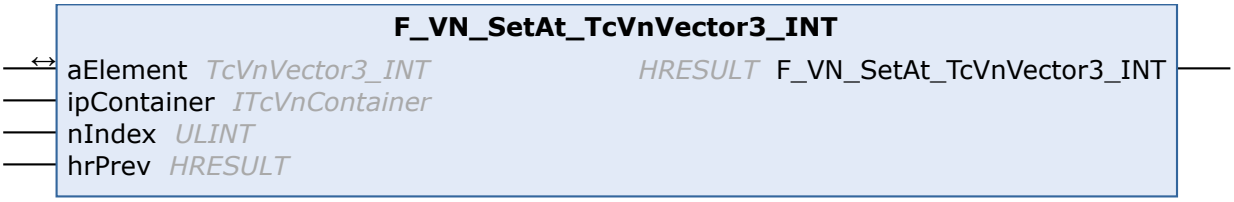
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.23 F_VN_SetAt_TcVnVector3_INT



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnVector3_INT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector3_INT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector3_INT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_INT [▶ 173]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

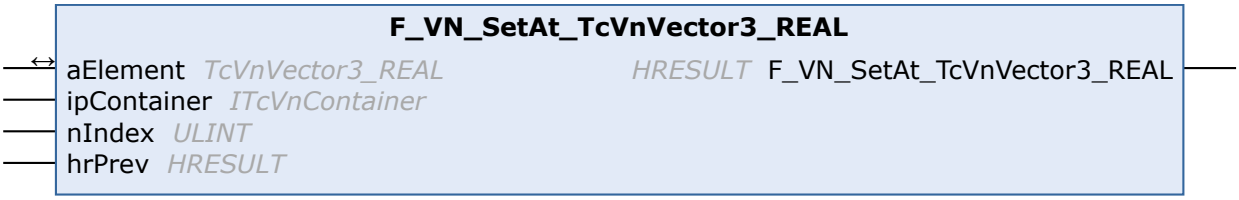
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.24 F_VN_SetAt_TcVnVector3_REAL



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnVector3_REAL : HRESULT
VAR_IN_OUT
    aElement     : TcVnVector3_REAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector3_REAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_REAL [▶ 173]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

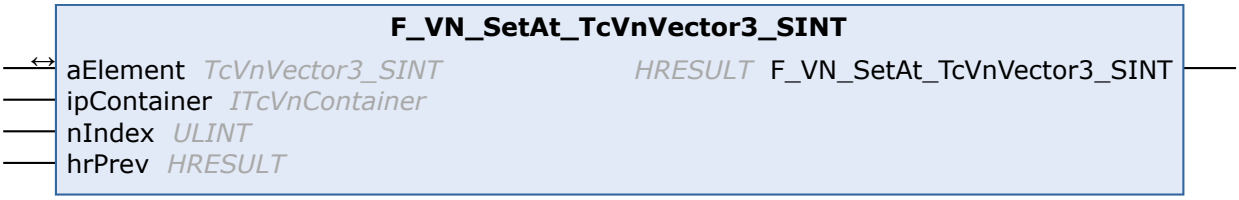
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.25 F_VN_SetAt_TcVnVector3_SINT



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnVector3_SINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector3_SINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector3_SINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_SINT [▶ 173]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

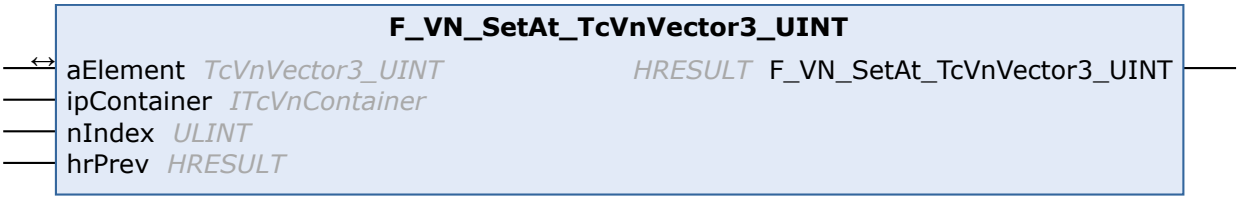
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.26 F_VN_SetAt_TcVnVector3_UINT



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnVector3_UINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector3_UINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector3_UINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_UINT [▶ 173]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

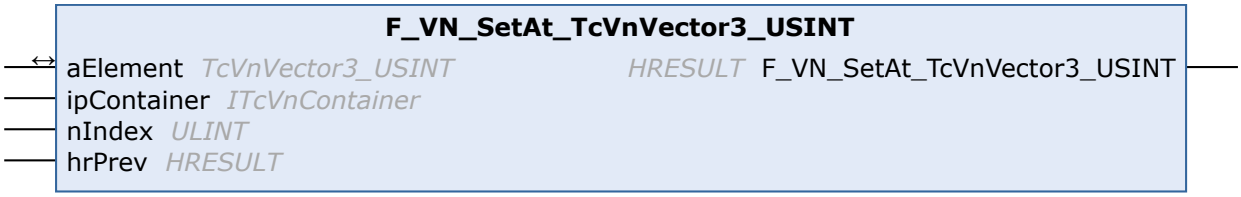
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.27 F_VN_SetAt_TcVnVector3_USINT



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnVector3_USINT : HRESULT
VAR_IN_OUT
    aElement     : TcVnVector3_USINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector3_USINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector3_USINT [▶ 173]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

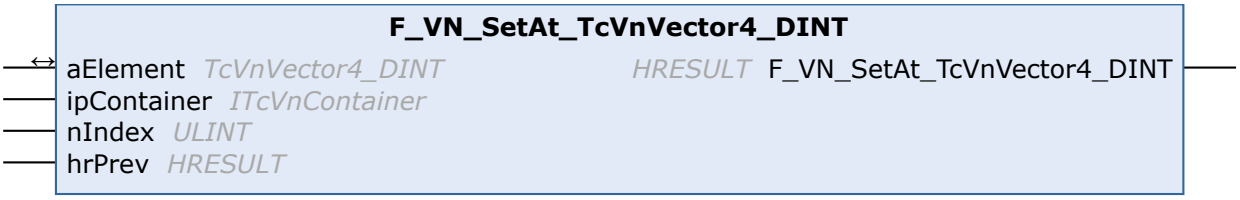
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.28 F_VN_SetAt_TcVnVector4_DINT



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnVector4_DINT : HRESULT
VAR_IN_OUT
    aElement     : TcVnVector4_DINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector4_DINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_DINT [▶ 173]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

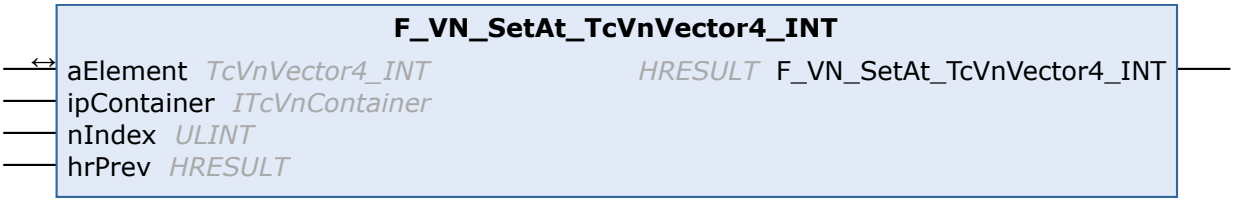
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.29 F_VN_SetAt_TcVnVector4_INT



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnVector4_INT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector4_INT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector4_INT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_INT [▶ 173]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

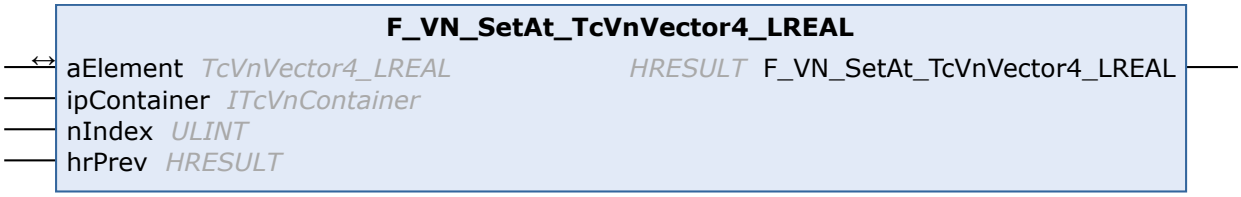
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.30 F_VN_SetAt_TcVnVector4_LREAL



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnVector4_LREAL : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector4_LREAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_LREAL [▶ 173]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

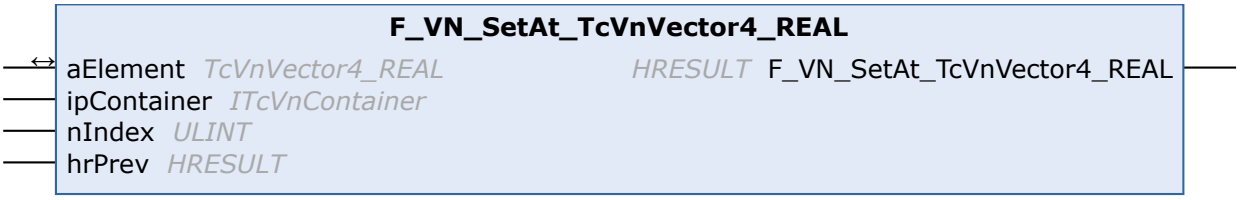
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.31 F_VN_SetAt_TcVnVector4_REAL



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnVector4_REAL : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector4_REAL;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector4_REAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_REAL [▶ 173]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

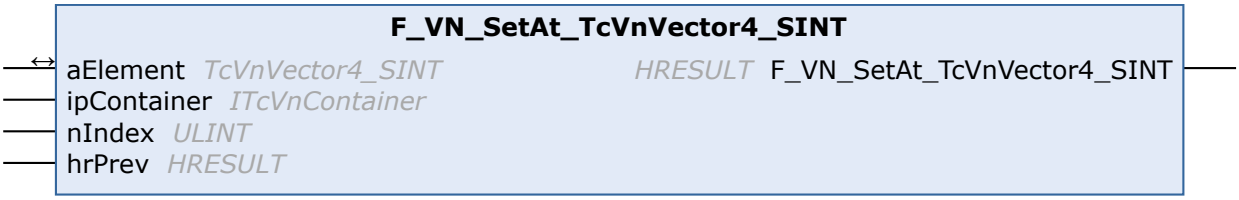
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.32 F_VN_SetAt_TcVnVector4_SINT



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnVector4_SINT : HRESULT
VAR_IN_OUT
    aElement     : TcVnVector4_SINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector4_SINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_SINT [▶ 173]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

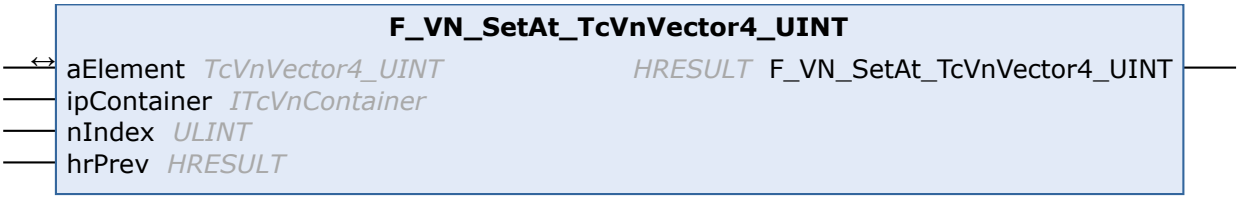
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.33 F_VN_SetAt_TcVnVector4_UINT



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnVector4_UINT : HRESULT
VAR_IN_OUT
    aElement     : TcVnVector4_UINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector4_UINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_UINT [▶ 173]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

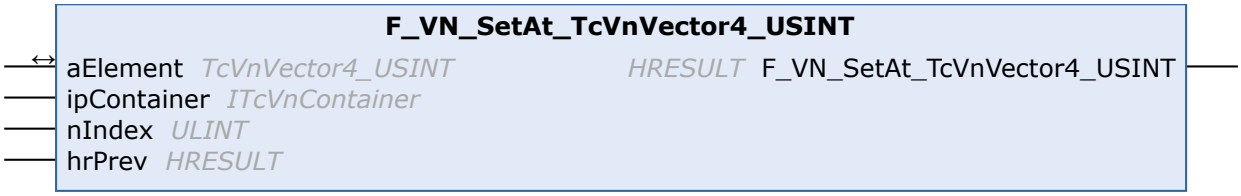
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.34 F_VN_SetAt_TcVnVector4_USINT



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_TcVnVector4_USINT : HRESULT
VAR_IN_OUT
    aElement    : TcVnVector4_USINT;
END_VAR
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with TcVnVector4_USINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aElement	TcVnVector4_USINT [▶ 173]	Element to set at the specified container position

Return value

HRESULT [▶ 150]

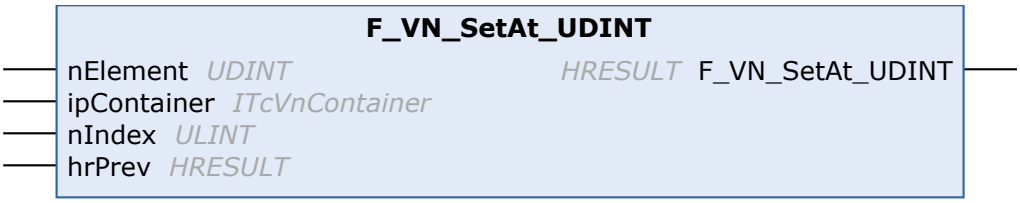
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.35 F_VN_SetAt_UDINT



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_UDINT : HRESULT
VAR_INPUT
    nElement    : UDINT;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nElement	UDINT	Element to set at the specified container position
ipContainer	ITcVnContainer [▶ 385]	Container with UDINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

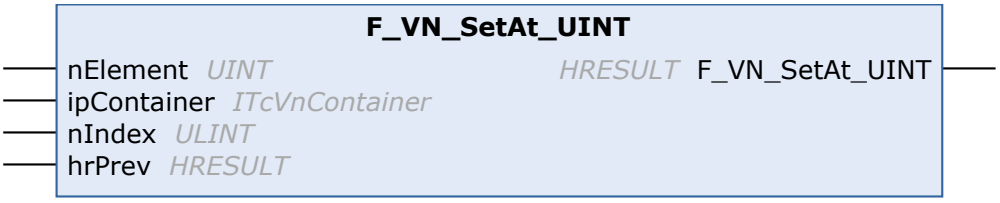
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.36 F_VN_SetAt_UINT



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_UINT : HRESULT
VAR_INPUT
    nElement    : UINT;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nElement	UINT	Element to set at the specified container position
ipContainer	ITcVnContainer [▶ 385]	Container with UINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

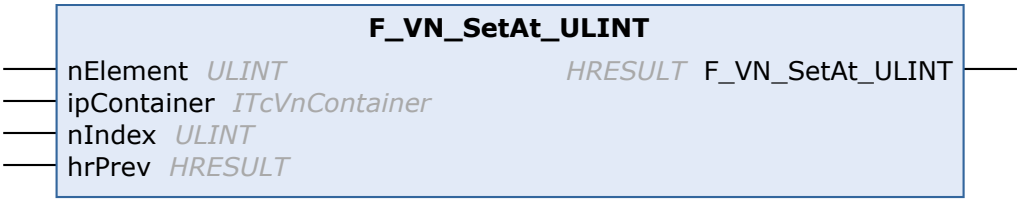
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.37 F_VN_SetAt_ULINT



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_ULINT : HRESULT
VAR_INPUT
    nElement    : ULINT;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nElement	ULINT	Element to set at the specified container position
ipContainer	ITcVnContainer [▶ 385]	Container with ULINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

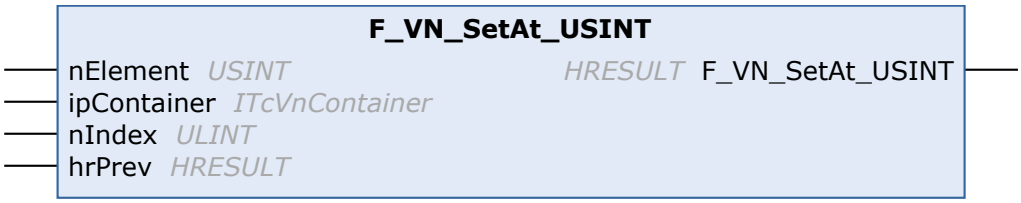
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.6.38 F_VN_SetAt_USINT



Sets the element at the specified index of the container.

Syntax

Definition:

```
FUNCTION F_VN_SetAt_USINT : HRESULT
VAR_INPUT
    nElement    : USINT;
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nElement	USINT	Element to set at the specified container position
ipContainer	ITcVnContainer [▶ 385]	Container with USINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULINT	Index
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.7 F_VN_AdvanceIterator

Advance an iterator by the specified offset.

Syntax

Definition:

```
FUNCTION F_VN_AdvanceIterator : HRESULT
VAR_INPUT
    ipIterator : ITcVnForwardIterator;
    nOffset    : LINT;
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipIterator	ITcVnForwardIterator [▶ 376]	Iterator to be advanced
nOffset	LINT	Offset, by which ipIterator is advanced
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_AdvanceIterator` inkrementiert oder dekrementiert einen Iterator [▶ 375] um einen bestimmten Offset `nOffset`.

Wenn der Offset `nOffset` immer `1` sein soll, kann ebenfalls die einfachere Funktion F_VN_IncrementIterator [▶ 791] verwendet werden.

Parameter

Iterator

Der Parameter `ipIterator` ist vom Typ `ITcVnForwardIterator`. Da `ITcVnBidirectionalIterator` und `ITcVnRandomAccessIterator` davon erben, können diese ebenfalls an die Funktion übergeben werden. Falls ein Vorwärts-Iterator dekrementiert werden soll, wird intern versucht, diesen in einen bidirektionalen Iterator umzuwandeln und diesen zu dekrementieren.

Offset

Das Offset `nOffset` legt fest, um wie viel der Iterator `ipIterator` inkrementiert oder dekrementiert wird. Bei `nOffset > 0` wird inkrementiert; bei `nOffset < 0` wird dekrementiert.

Verwandte Funktionen

- F_VN_IncrementIterator [▶ 791]
- F_VN_AdvanceIterator [▶ 768]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.8 F_VN_CheckIfEmpty

Checks if the container is empty. (Alternatively use interface method .CheckIfEmpty.)

Syntax

Definition:

```
FUNCTION F_VN_CheckIfEmpty : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.9 F_VN_CheckIfIteratorIsAtEnd

Checks if the iterator points to the past-the-end element. (Alternatively use interface method .CheckIfEnd.)

Syntax

Definition:

```
FUNCTION F_VN_CheckIfIteratorIsAtEnd : HRESULT
VAR_INPUT
    ipIterator : ITcVnForwardIterator;
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipIterator	ITcVnForwardIterator [▶ 376]	Iterator
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.10 F_VN_ConvertContainerType

Converts a container to another type (Struct element types are not supported).

Syntax

Definition:

```
FUNCTION F_VN_ConvertContainerType : HRESULT
VAR_INPUT
    ipSrcContainer  : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    nDestTypeGuid   : GUID;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer	ITcVnContainer [▶ 385]	Source container
ipDestContainer	Reference To ITcVnContainer [▶ 385]	Returns the converted container
nDestTypeGuid	GUID	Specifies the destination container type for the conversion
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_ConvertContainerType` konvertiert einen Container in einen anzugebenden Container-Typ [▶ 171].

Zur Festlegung des Container-Typs über den Parameter `nTypeGuid` können die globalen ContainerType [▶ 171]-Konstanten verwendet werden.

HRESULT

70E	INCOMPATIBLE	Als Eingangscontainer und als Ergebniscontainer wird derselbe Interface Pointer verwendet oder als Interface Pointer wird ein Container verwendet, der einen anderen Typ als den in `nDestTypeGuid` angegebenen hat.

Anwendung

Ein typischer Anwendungsfall ist zum Beispiel die Konvertierung eines Containers, um diesen in ein Bild einzeichnen zu können. Eine Kontur von Typ `ContainerType_Vector_TcVnPoint2_REAL`, wie sie unter anderem von der Funktion F_VN_LocateEllipseExp [▶ 1541] zurückgegeben wird, kann in den Typ `ContainerType_Vector_TcVnPoint2_DINT` konvertiert werden, um diese mit der Funktion F_VN_DrawContours [▶ 1040] in ein Bild einzeichnen zu können:

```
hr := F_VN_LocateEllipseExp(ipImage, stEllipse, […], ipContourPoints, hr);
hr := F_VN_ConvertContainerType(
    ipSrcContainer  :=  ipContourPoints,
```

```
    ipDestContainer :=  ipContourToDraw,
    nDestTypeGuid   :=  ContainerType_Vector_TcVnPoint2_DINT,
    hr
);
hr := F_VN_DrawContours(ipContourToDraw, -1, ipImageDisp, aBlue, 5, hr);
```

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.11 F_VN_CopyContainer

Copys a container.

Syntax

Definition:

```
FUNCTION F_VN_CopyContainer : HRESULT
VAR_INPUT
    ipSrcContainer  : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer	ITcVnContainer [▶ 385]	Source container
ipDestContainer	Reference To ITcVnContainer [▶ 385]	Destination container (same type ID as ipSrcContainer; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CopyContainer` kopiert nicht nur den Interface Pointer [▶ 146] des Containers [▶ 163], sondern erstellt eine tiefe Kopie des Containers.

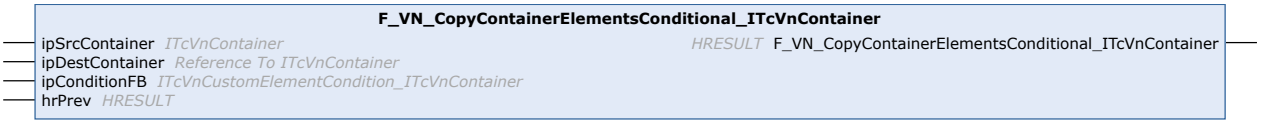
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.12 F_VN_CopyContainerElementsConditional_ITcVnContainer



Copys container elements to a new container, depending on a custom condition.

Syntax

Definition:

```
FUNCTION F_VN_CopyContainerElementsConditional_ITcVnContainer : HRESULT
VAR_INPUT
    ipSrcContainer  : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    ipConditionFB   : ITcVnCustomElementCondition_ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer	ITcVnContainer [▶ 385]	Source container
ipDestContainer	Reference To ITcVnContainer [▶ 385]	Returns a container with elements, that match the condition
ipConditionFB	ITcVnCustomElementCondition_ITcVnContainer [▶ 261]	Custom condition
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CopyContainerElementsConditional` filtert Elemente von mehrdimensionalen Containern [▶ 163]. Die Filter-Bedingung ist durch einen Funktionsblock mit dem Interface ITcVnCustomElementCondition_ITcVnContainer [▶ 261] anzugeben.

Parameter

Eingangscontainer

Der Eingangscontainer `ipSrcContainer` muss ein mehrdimensionaler Container sein; d. h. der Container muss als Elemente Sub-Container enthalten. Somit kommen alle Container vom Typ `ContainerType_Vector_Vector_<...>` in Frage.

Ergebniscontainer

Der Ergebniscontainer `ipDestContainer` enthält alle Sub-Container, die die Bedingung in `ipConditionFB` erfüllen.

Filter-Bedingung

Die Filter-Bedingung muss als Methode `Condition` eines Funktionsblocks definiert werden. Damit dieser Funktionsblock als Parameter `ipConditionFB` an die Funktion übergeben werden kann, muss er das Interface ITcVnCustomElementCondition_ITcVnContainer [▶ 261] implementieren.

Beispiele

- Selektieren von Container-Elementen [▶ 2840]

Verwandte Funktionen

- F_VN_CopyContainerElementsConditional_ITcVnContainer [▶ 773]
- F_VN_CopyContainerElementsConditional_ITcVnForwardIterator [▶ 774]

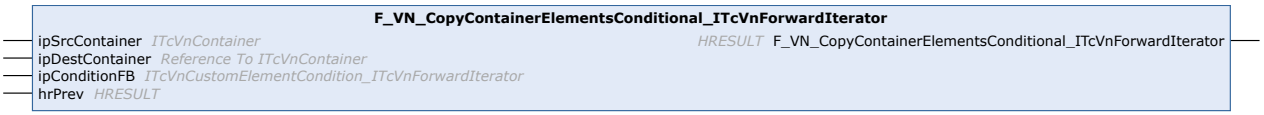
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.13 F_VN_CopyContainerElementsConditional_ITcVnForwardIterator



Copys container elements to a new container, depending on a custom condition.

Syntax

Definition:

```
FUNCTION F_VN_CopyContainerElementsConditional_ITcVnForwardIterator : HRESULT
VAR_INPUT
    ipSrcContainer  : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    ipConditionFB   : ITcVnCustomElementCondition_ITcVnForwardIterator;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer	ITcVnContainer [▶ 385]	Source container
ipDestContainer	Reference To ITcVnContainer [▶ 385]	Returns a container with elements, that match the condition
ipConditionFB	ITcVnCustomElementCondition_ITcVnForwardIterator [▶ 262]	Custom condition
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CopyContainerElementsConditional_ITcVnForwardIterator` filtert Elemente von mehrdimensionalen Containern [▶ 163]. Die Filter-Bedingung ist durch einen Funktionsblock mit dem Interface ITcVnCustomElementCondition_ITcVnForwardIterator [▶ 262] anzugeben.

Im Gegensatz zu F_VN_CopyContainerElementsConditional_ITcVnContainer [▶ 773] werden nicht die Sub-Container an die Bedingungsmethode `Condition` übergeben, sondern Iteratoren, die jeweils auf das erste Element des Sub-Containers zeigen.

Parameter

Eingangscontainer

Der Eingangscontainer `ipSrcContainer` muss ein mehrdimensionaler Container sein; d. h. der Container muss als Elemente Sub-Container enthalten. Somit kommen alle Container vom Typ `ContainerType_Vector_Vector_<…>` in Frage.

Ergebniscontainer

Der Ergebniscontainer `ipDestContainer` enthält alle Sub-Container, die die Bedingung in `ipConditionFB` erfüllen.

Filter-Bedingung

Die Filter-Bedingung muss als Methode `Condition` eines Funktionsblocks definiert werden. Damit dieser Funktionsblock als Parameter `ipConditionFB` an die Funktion übergeben werden kann, muss er das Interface ITcVnCustomElementCondition_ITcVnForwardIterator [▶ 262] implementieren.

Beispiele

- Selektieren von Container-Elementen [▶ 2840]

Verwandte Funktionen

- F_VN_CopyContainerElementsConditional_ITcVnContainer [▶ 773]
- F_VN_CopyContainerElementsConditional_ITcVnForwardIterator [▶ 774]

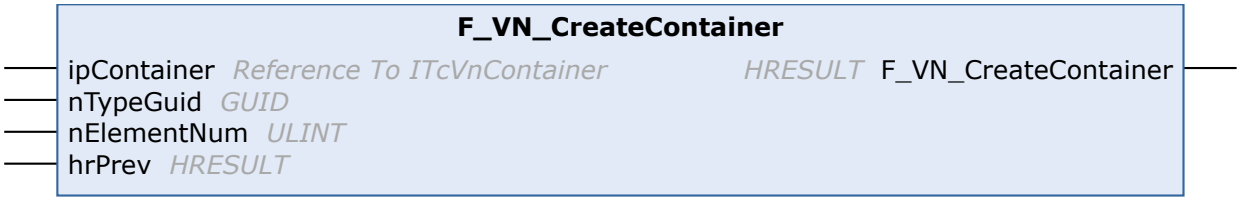
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.14 F_VN_CreateContainer



Creates a container with type GUID nTypeGuid and return its container interface.

Syntax

Definition:

```
FUNCTION F_VN_CreateContainer : HRESULT
VAR_INPUT
    ipContainer : Reference To ITcVnContainer;
    nTypeGuid   : GUID;
    nElementNum : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	Reference To ITcVnContainer [▶ 385]	Returns the created container
nTypeGuid	GUID	Type GUID of the container to be created
nElementNum	ULINT	Number of elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Zur Festlegung des Container-Typs über den Parameter `nTypeGuid` können die globalen ContainerType [▶ 171]-Konstanten verwendet werden.

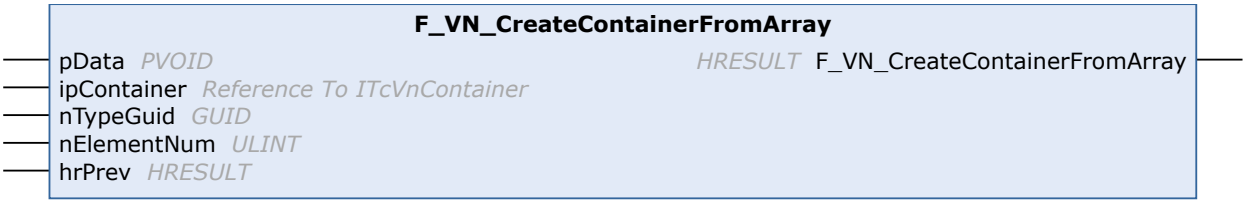
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.15 F_VN_CreateContainerFromArray



Creates a container with type GUID nTypeGuid, initialize it with provided data and return its container interface. Only supported for basic container types, i.e. not for containers of containers.

Syntax

Definition:

```
FUNCTION F_VN_CreateContainerFromArray : HRESULT
VAR_INPUT
    pData       : PVOID;
    ipContainer : Reference To ITcVnContainer;
    nTypeGuid   : GUID;
```

```
    nElementNum : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
pData	PVOID	Pointer to the data array. Make sure that the array contains at least nElementNum elements and that the array element type matches the container element type.
ipContainer	Reference To ITcVnContainer [▶ 385]	Returns the created container
nTypeGuid	GUID	Type GUID of the container to be created
nElementNum	ULINT	Number of elements to copy from pData
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Zur Festlegung des Container-Typs über den Parameter `nTypeGuid` können die globalen ContainerType [▶ 171]-Konstanten verwendet werden.

Da die Daten des Arrays nur über einen Pointer angegeben werden, ist eine Typüberprüfung innerhalb der Funktion nicht möglich. Beachten Sie daher folgende Warnung:

HINWEIS

Speicherzugriff

Es muss sichergestellt sein, dass der Typ und Länge des Containers mit der Größe des Speicherbereichs am Pointer übereinstimmen. Anderenfalls kann dies zu einem Systemabsturz oder invaliden Daten führen.

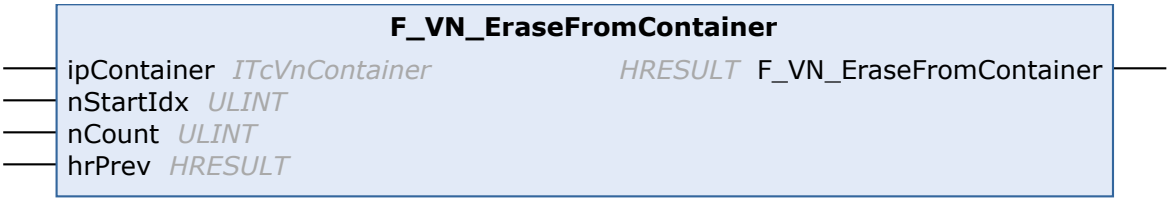
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.16 F_VN_EraseFromContainer



Erase elements from a container.

Syntax

Definition:

```
FUNCTION F_VN_EraseFromContainer : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nStartIdx   : ULINT;
    nCount      : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container from which to erase elements
nStartIdx	ULINT	Index of the first element to erase
nCount	ULINT	Number of elements to erase
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.17 F_VN_ExportContainer

Export the container elements into a buffer (e.g. an array). Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_ExportContainer : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    pBuffer     : PVOID;
    nBufferSize : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
pBuffer	PVOID	Buffer to store the container elements (Make sure to allocate enough memory! The required size in bytes can be determined using the function F_VN_ExportContainerSize)
nBufferSize	ULINT	Size of the buffer memory in bytes
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_ExportContainer` exportiert einen Container in einen Datenbuffer. Stellen Sie sicher, dass an der Stelle des Pointers genügend Speicher allokiert ist, um die Daten des Containers hineinzuschreiben! Die benötigte Größe in Bytes kann mit der Funktion F_VN_ExportContainerSize [▶ 781] festgestellt werden. Den Speicher können Sie z. B. allokieren, indem Sie ein Array mit entsprechender Größe anlegen.

Anwendung

Das Exportieren eines Containers des Typs `Vector_REAL` mit `10` Elementen sieht z. B. so aus:

```
VAR
    aArray: ARRAY [0..9] OF REAL;
    ipContainer: ITcVnContainer;
    nBufferSize: ULINT;
END_VAR

hr := F_VN_ExportContainerSize(ipContainer, nBufferSize, hr);
IF nBufferSize = SIZEOF(aArray) THEN
    hr := F_VN_ExportContainer(
        ipContainer:=ipContainer,
        pBuffer:=ADR(aArray),
        nBufferSize:=nBufferSize,
        hr
    );
END_IF
```

HINWEIS

Speicherzugriff

Es muss sichergestellt sein, dass der Typ und Länge des Containers mit der Größe des Speicherbereichs am Pointer übereinstimmen. Anderenfalls kann dies zu einem Systemabsturz oder invaliden Daten führen.

Verwandte Funktionen

- F_VN_ExportContainer [▶ 778]
- F_VN_ExportSubContainer [▶ 782]
- F_VN_ExportContainer_String [▶ 780]
- F_VN_ExportSubContainer_String [▶ 783]
- F_VN_ExportContainerSize [▶ 781]
- F_VN_ExportSubContainerSize [▶ 784]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.18 F_VN_ExportContainer_String

Export the container elements into a string. Only possible for containers of type ContainerType_String_SINT.

Syntax

Definition:

```
FUNCTION F_VN_ExportContainer_String : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    sText       : STRING;
    nMaxLength  : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container of type ContainerType_String_SINT
sText	STRING	Make sure to choose a sufficient STRING size! The required size can be determined using the function VnExportContainerSize.
nMaxLength	ULINT	Maximum string length to export (including 0 termination). If the container content is longer, the string is cut off at nMaxLength - 1 and 0 termination is appended. In this case, S_FALSE is returned.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Verwandte Funktionen

- F_VN_ExportContainer [▶ 778]
- F_VN_ExportSubContainer [▶ 782]
- F_VN_ExportContainer_String [▶ 780]
- F_VN_ExportSubContainer_String [▶ 783]
- F_VN_ExportContainerSize [▶ 781]
- F_VN_ExportSubContainerSize [▶ 784]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.19 F_VN_ExportContainerSize

Determine the required buffer size in bytes to store all container elements (number_of_Elements * size_per_Element). Only possible for containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_ExportContainerSize : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nBufferSize : ULINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nBufferSize	ULINT	Output parameter containing the required buffer size

Return value

HRESULT [▶ 150]

Verwandte Funktionen

- F_VN_ExportContainer [▶ 778]
- F_VN_ExportSubContainer [▶ 782]
- F_VN_ExportContainer_String [▶ 780]
- F_VN_ExportSubContainer_String [▶ 783]
- F_VN_ExportContainerSize [▶ 781]
- F_VN_ExportSubContainerSize [▶ 784]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.20 F_VN_ExportSubContainer

Export the container elements of a sub-container into a buffer (e.g. an array). Only possible for 2-dimensional containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_ExportSubContainer : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    pBuffer     : PVOID;
    nBufferSize : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nIndex	ULINT	Index of the requested element
pBuffer	PVOID	Buffer to store the container elements (Make sure to allocate enough memory! The required size in bytes can be determined using the function F_VN_ExportSubContainerSize)
nBufferSize	ULINT	Size of the buffer memory in bytes
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Verwandte Funktionen

- F_VN_ExportContainer [▶ 778]
- F_VN_ExportSubContainer [▶ 782]
- F_VN_ExportContainer_String [▶ 780]
- F_VN_ExportSubContainer_String [▶ 783]
- F_VN_ExportContainerSize [▶ 781]

- F_VN_ExportSubContainerSize [▶ 784]

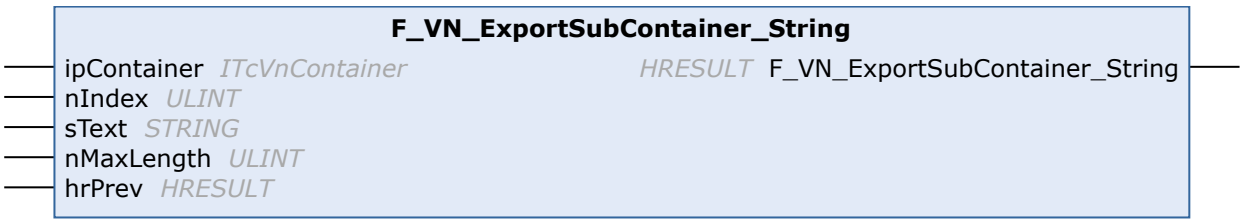
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.21 F_VN_ExportSubContainer_String



Export the container elements of a sub-container into a string. Only possible for 2-dimensional containers of type ContainerType_Vector_String_SINT.

Syntax

Definition:

```
FUNCTION F_VN_ExportSubContainer_String : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
    sText       : STRING;
    nMaxLength  : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container of type ContainerType_Vector_String_SINT
nIndex	ULINT	Index of the requested element
sText	STRING	Make sure to choose a sufficient STRING size! The required size can be determined using the function VnExportSubContainerSize.
nMaxLength	ULINT	Maximum string length to export (including 0 termination). If the container content is longer, the string is cut off at nMaxLength - 1 and 0 termination is appended. In this case, S_FALSE is returned.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Anwendung

Die Funktion `F_VN_ExportSubContainer` wird z. B. zum Erlangen der Ergebnistexte beim Code Reading [▶ 863] verwendet:

```
hr := F_VN_ReadPharmaCode(ipImage, ipDecodedText, hr);
IF hr = S_OK THEN
    hr := F_VN_ExportSubContainer_String(ipDecodedText, 0, sCodeAsString, 255, hr);
    // sCodeAsString contains the result
END_IF
```

Verwandte Funktionen

- F_VN_ExportContainer [▶ 778]
- F_VN_ExportSubContainer [▶ 782]
- F_VN_ExportContainer_String [▶ 780]
- F_VN_ExportSubContainer_String [▶ 783]
- F_VN_ExportContainerSize [▶ 781]
- F_VN_ExportSubContainerSize [▶ 784]

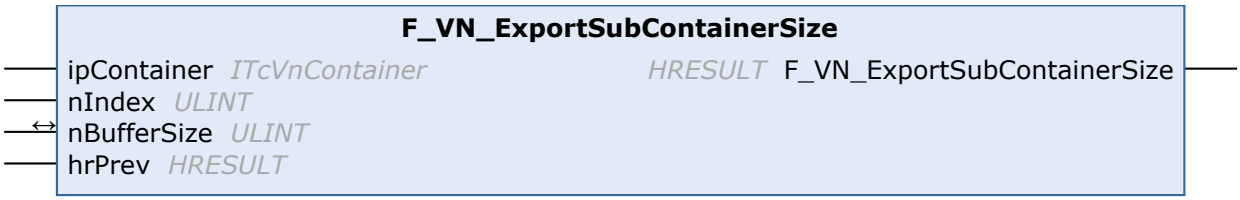
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.22 F_VN_ExportSubContainerSize



Determine the required buffer size in bytes to store all container elements of a sub-container (number_of_Elements * size_per_Element). Only possible for 2-dimensional containers with basic elements.

Syntax

Definition:

```
FUNCTION F_VN_ExportSubContainerSize : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nIndex      : ULINT;
END_VAR
VAR_IN_OUT
    nBufferSize : ULINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container with basic elements
nIndex	ULINT	Index of the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nBufferSize	ULINT	Output parameter containing the required buffer size

Return value

HRESULT [▶ 150]

Verwandte Funktionen

- F_VN_ExportContainer [▶ 778]
- F_VN_ExportSubContainer [▶ 782]
- F_VN_ExportContainer_String [▶ 780]
- F_VN_ExportSubContainer_String [▶ 783]
- F_VN_ExportContainerSize [▶ 781]
- F_VN_ExportSubContainerSize [▶ 784]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.23 F_VN_ExtractContainerRange

Copy the specified range of the source container into the destination container. (If the destination container already contains elements, they will be removed.)

Syntax

Definition:

```
FUNCTION F_VN_ExtractContainerRange : HRESULT
VAR_INPUT
    ipSrcContainer  : ITcVnContainer;
    ipDestContainer : Reference To ITcVnContainer;
    nFirstIdx       : ULINT;
    nLastIdx        : ULINT;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer	ITcVnContainer [▶ 385]	Source container
ipDestContainer	Reference To ITcVnContainer [▶ 385]	Destination container
nFirstIdx	ULINT	Index of the first element to copy
nLastIdx	ULINT	Index of the last element to copy
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

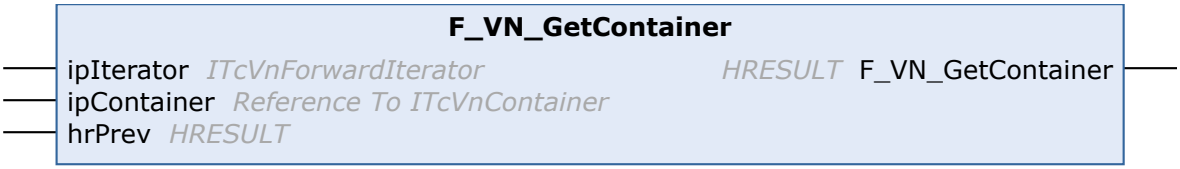
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.24 F_VN_GetContainer



Gets a pointer to the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). (Alternatively use interface method .GetContainer.)

Syntax

Definition:

```
FUNCTION F_VN_GetContainer : HRESULT
VAR_INPUT
    ipIterator  : ITcVnForwardIterator;
    ipContainer : Reference To ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipIterator	ITcVnForwardIterator [▶ 376]	Iterator
ipContainer	Reference To ITcVnContainer [▶ 385]	Returns the container interface
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Tiefe Kopie

Diese Funktion erstellt eine tiefe Kopie des Container-Elements. D. h., wenn Sie das Element im Container verändern wollen, müssen Sie die Änderungen separat in den Container zurückschreiben.

Wenn Sie gleichzeitig mit dem Zugreifen auf das Container-Element auch den Iterator inkrementieren möchten, steht dafür die Expert-Variante F_VN_GetContainerExp [▶ 787] zur Verfügung.

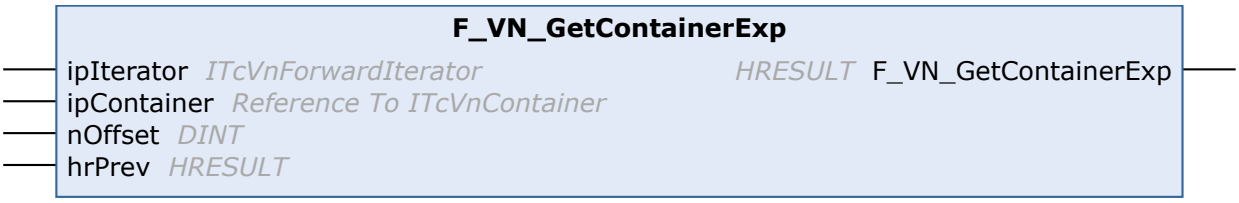
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.25 F_VN_GetContainerExp



Gets a pointer to the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). Additionally, an offset to increment or decrement the iterator afterwards can be provided. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_GetContainerExp : HRESULT
VAR_INPUT
    ipIterator  : ITcVnForwardIterator;
    ipContainer : Reference To ITcVnContainer;
    nOffset     : DINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipIterator	ITcVnForwardIterator [▶ 376]	Iterator
ipContainer	Reference To ITcVnContainer [▶ 385]	Returns the container interface
nOffset	DINT	Offset to increment (>0) or decrement (<0) the iterator afterwards
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_GetContainerExp` ist die Expert-Variante von F_VN_GetContainer [▶ 786]. Sie hilft Ihnen, wenn Sie gleichzeitig mit dem Zugreifen auf ein Container-Element mittels Iterator auch diesen Iterator inkrementieren möchten. Setzen Sie den Parameter `nOffset` für ein normales, sequenzielles Inkrementieren auf den Wert `1`:

```
hr := F_VN_GetContainerExp(ipIterator, ipElement, 1, hr);
```

Mit der Standard-Funktion würden Sie dafür zwei Funktionsaufrufe benötigen:

```
hr := F_VN_GetContainer(ipIterator, ipElement, hr);
hr := F_VN_IncrementIterator(ipIterator, hr);
```

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.26 F_VN_GetForwardIterator

Gets a forward iterator for the container. (Alternatively use interface method .GetForwardIterator.)

Syntax

Definition:

```
FUNCTION F_VN_GetForwardIterator : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    ipIterator  : Reference To ITcVnForwardIterator;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container
ipIterator	Reference To ITcVnForwardIterator [▶ 376]	Returns the iterator interface
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

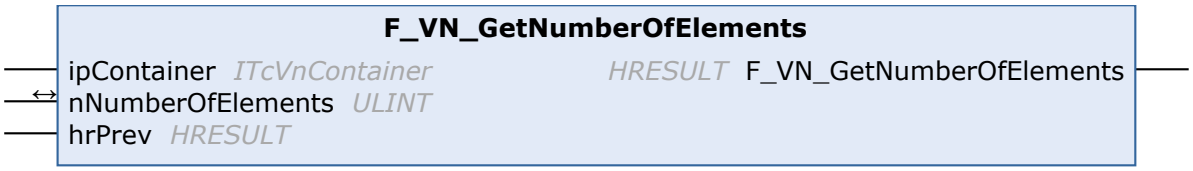
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.27 F_VN_GetNumberOfElements



Gets the number of elements in the container. (Alternatively use interface method .getElementNum.)

Syntax

Definition:

```
FUNCTION F_VN_GetNumberOfElements : HRESULT
VAR_INPUT
    ipContainer       : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nNumberOfElements : ULINT;
END_VAR
VAR_INPUT
    hrPrev            : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nNumberOfElements	ULINT	Returns the number of elements in ipContainer

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Diese Funktion sollte nicht genutzt werden, um über die Elemente eines Containers zu Iterieren. Stattdessen sollten Sie mit Iteratoren auf die Elemente zugreifen [▶ 164]. Insbesondere wird von folgendem Codekonstrukt abgeraten:

```
hr := F_VN_GetNumberOfElements(ipContainer, ipNumber, hr);
FOR i:=0 TO (ipNumber-1) DO
    // access container elements
END_FOR
```

Da `ipNumber` ein `UDINT` (natürliche Ganzzahl) ist, würde ein Rückgabewert von `ipNumber = 0` einen Unterlauf verursachen.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.28 F_VN_GetRandomAccessIterator

Gets a random access iterator for the container. (Alternatively use interface method .GetRandomAccessIterator.)

Syntax

Definition:

```
FUNCTION F_VN_GetRandomAccessIterator : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    ipIterator  : Reference To ITcVnRandomAccessIterator;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container
ipIterator	Reference To ITcVnRandomAccessIterator [▶ 382]	Returns the iterator interface
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

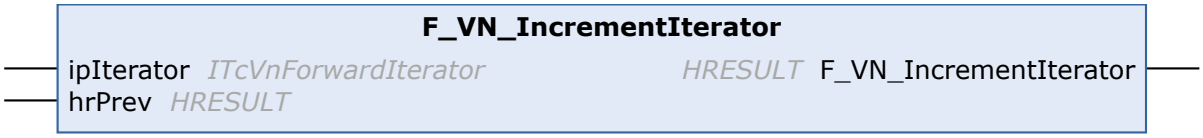
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.29 F_VN_IncrementIterator



Increment the iterator. (Alternatively use interface method .Increment.)

Syntax

Definition:

```
FUNCTION F_VN_IncrementIterator : HRESULT
VAR_INPUT
    ipIterator : ITcVnForwardIterator;
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipIterator	ITcVnForwardIterator [▶ 376]	Iterator
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_IncrementIterator` inkrementiert einen Iterator [▶ 375] um ein Element. Wenn Sie den Iterator stattdessen dekrementieren oder um mehrere Positionen inkrementieren möchten, nutzen Sie die Funktion F_VN_AdvanceIterator [▶ 768].

Parameter

Iterator

Der Parameter `ipIterator` ist vom Typ `ITcVnForwardIterator`. Da `ITcVnBidirectionalIterator` und `ITcVnRandomAccessIterator` davon erben, können diese ebenfalls an die Funktion übergeben werden. Falls ein Vorwärts-Iterator dekrementiert werden soll, wird intern versucht, diesen in einen bidirektionalen Iterator umzuwandeln und diesen zu dekrementieren.

Verwandte Funktionen

- F_VN_IncrementIterator [▶ 791]
- F_VN_AdvanceIterator [▶ 768]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.30 F_VN_IteratorDistance

Computes the distance between 2 iterators.

Syntax

Definition:

```
FUNCTION F_VN_IteratorDistance : HRESULT
VAR_INPUT
    ipIteratorFirst : ITcVnForwardIterator;
    ipIteratorLast  : ITcVnForwardIterator;
END_VAR
VAR_IN_OUT
    nDistance       : LINT;
END_VAR
VAR_INPUT
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipIteratorFirst	ITcVnForwardIterator [▶ 376]	First iterator
ipIteratorLast	ITcVnForwardIterator [▶ 376]	Last iterator
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nDistance	LINT	Returns the computed distance

Return value

HRESULT [▶ 150]

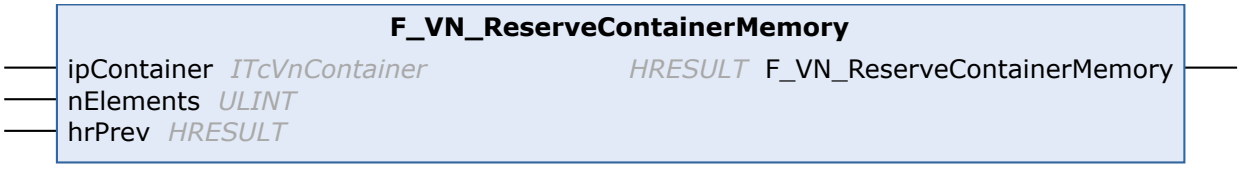
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.31 F_VN_ReserveContainerMemory



Reserve container memory (call with maximum required number of elements before manually appending elements for better performance)

Syntax

Definition:

```
FUNCTION F_VN_ReserveContainerMemory : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    nElements   : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container for which to reserve the memory
nElements	ULINT	Number of elements for which the container should reserve memory
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

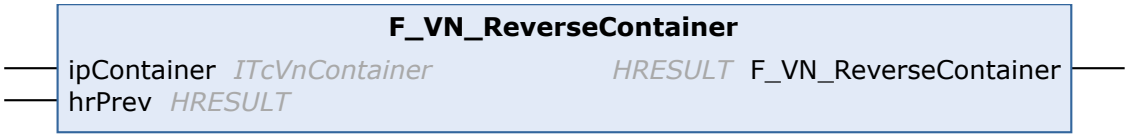
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.32 F_VN_ReverseContainer



Reverse the container elements.

Syntax

Definition:

```
FUNCTION F_VN_ReverseContainer : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Container to be reversed
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.33 F_VN_SetContainer

Sets the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use interface method .SetContainer.)

Syntax

Definition:

```
FUNCTION F_VN_SetContainer : HRESULT
VAR_INPUT
    ipIterator  : ITcVnForwardIterator;
    ipContainer : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipIterator	ITcVnForwardIterator [▶ 376]	Iterator
ipContainer	ITcVnContainer [▶ 385]	Container interface of which the content is to be assigned to the current element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

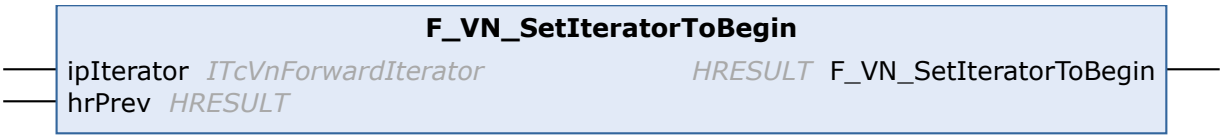
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.4.34 F_VN_SetIteratorToBegin



Sets the iterator to the first element of the container. (Alternatively use interface method .SetToBegin.)

Syntax

Definition:

```
FUNCTION F_VN_SetIteratorToBegin : HRESULT
VAR_INPUT
    ipIterator : ITcVnForwardIterator;
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipIterator	ITcVnForwardIterator [▶ 376]	Iterator
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5 Basic Image Operations

Die Gruppe enthält Funktionen zur Handhabung von Bildern [▶ 159].

Funktionen

Erstellung von Bildern

- F_VN_ConvertElementType(Exp) [▶ 798]

- F_VN_CreateAssociatedImage [▶ 804]
- F_VN_CreateEmptyImage [▶ 805]
- F_VN_CreateImage [▶ 806]
- F_VN_CreateImageAndSetPixels [▶ 807]
- F_VN_CreateImageFromArray [▶ 808]
- F_VN_CreateNDimensionalImage [▶ 837]
- F_VN_CreateNDimensionalImageAndSetPixels [▶ 838]
- F_VN_CreateNDimensionalImageFromArray [▶ 839]

Bild-Informationen

- F_VN_GetImageHeight [▶ 819]
- F_VN_GetImageInfo [▶ 820]
- F_VN_GetImageWidth [▶ 820]
- F_VN_GetLayoutInfo [▶ 840]
- F_VN_GetPixelFormat [▶ 822]

Kopieren von Bildern

- F_VN_CopyImage [▶ 799]
- F_VN_CopyImageRegion [▶ 801]
- F_VN_CopyImageRegionToRegion [▶ 802]

Pixel-Manipulation

- F_VN_GetPixel [▶ 821]
- F_VN_SetPixel [▶ 827]
- F_VN_SetPixels(Exp) [▶ 828]

Region of Interest

- F_VN_GetRoi [▶ 823]
- F_VN_ResetRoi [▶ 825]
- F_VN_SetRoi [▶ 830]
- F_VN_SetRoi_TcVnRectangle_DINT [▶ 831]
- F_VN_SetRoi_TcVnRectangle_UDINT [▶ 832]

Bildinhalt in Speicher-Bereich exportieren

- F_VN_ExportImage [▶ 809]
- F_VN_ExportImageAsBmp(Exp) [▶ 810]
- F_VN_ExportImageSize [▶ 814]
- F_VN_ExportImageAsBmpSize(Exp) [▶ 812]

Bilder fusionieren

- F_VN_FuseImages [▶ 815]
- F_VN_FuseImagesArray [▶ 817]

Handhabung von Bilder-Kanälen

- F_VN_CombineImageChannels [▶ 797]
- F_VN_GetImageChannel [▶ 818]
- F_VN_MixImageChannels [▶ 824]

- F_VN_SetImageChannel [▶ 826]
- F_VN_SplitImageChannels [▶ 834]

Anzeigen von Bildern

- F_VN_CopyIntoDisplayableImage [▶ 804]
- F_VN_TransformIntoDisplayableImage(Exp) [▶ 835]

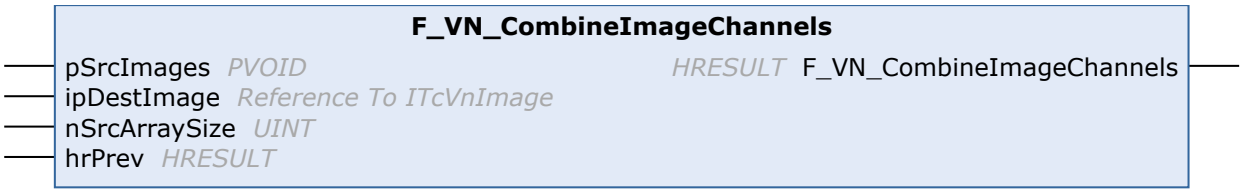
Sonstiges

- F_VN_PadImageBorder [▶ 841]
- F_VN_ReinterpretUnsupportedImage [▶ 843]

Beispiele

Basic Image Operations [▶ 2842]

6.1.4.5.1 F_VN_CombineImageChannels



Combines several single-channel images into one multi-channel image.

Syntax

Definition:

```
FUNCTION F_VN_CombineImageChannels : HRESULT
VAR_INPUT
    pSrcImages    : PVOID;
    ipDestImage   : Reference To ITcVnImage;
    nSrcArraySize : UINT;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
pSrcImages	PVOID	Pointer to an array of single-channel source images
ipDestImage	Reference To ITcVnImage [▶ 426]	Multi-channel destination image (An appropriate destination image will be created if required.)
nSrcArraySize	UINT	pSrcImages array size
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

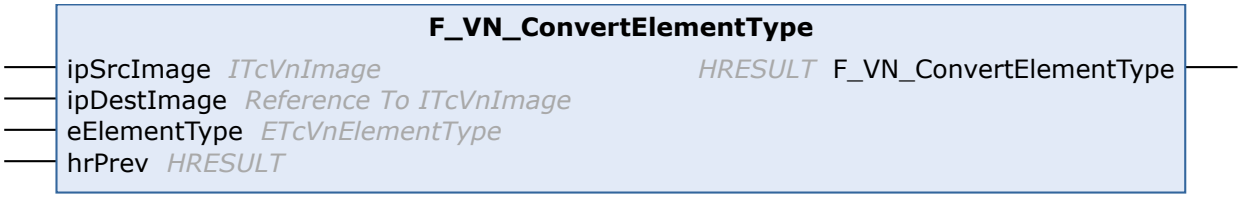
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.2 F_VN_ConvertElementType



Converts an image to another element type.

Syntax

Definition:

```
FUNCTION F_VN_ConvertElementType : HRESULT
VAR_INPUT
    ipSrcImage   : ITcVnImage;
    ipDestImage  : Reference To ITcVnImage;
    eElementType : ETcVnElementType;
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
eElementType	ETcVnElementType [▶ 210]	Desired element type of the destination image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

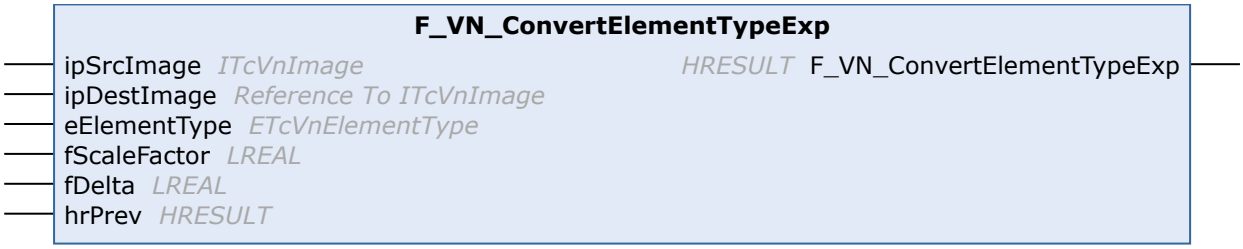
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.3 F_VN_ConvertElementTypeExp



Converts an image to another element type. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_ConvertElementTypeExp : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipDestImage   : Reference To ITcVnImage;
    eElementType  : ETcVnElementType;
    fScaleFactor  : LREAL;
    fDelta        : LREAL;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
eElementType	ETcVnElementType [▶ 210]	Desired element type of the destination image
fScaleFactor	LREAL	Scale factor for the pixel values
fDelta	LREAL	Value that is added to the scaled pixel values
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.4 F_VN_CopyImage

Creates a deep copy of an image.

Syntax

Definition:

```
FUNCTION F_VN_CopyImage : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CopyImage` erstellt eine tiefe Kopie des Eingangsbilds `ipSrcImage`. Alternativ kann mit der Funktion F_VN_CopyImageRegion [▶ 801] eine Kopie eines Bildausschnittes erzeugt werden.

Rechenzeitoptimierung

Prüfen Sie in Bezug auf Rechenzeitoptimierung jede Bildkopie auf ihre Notwendigkeit.

Verwandte Funktionen

- F_VN_CopyImage [▶ 799] zum Kopieren des gesamten Bildes
- F_VN_CopyImageRegion [▶ 801] zum Kopieren eines Bildbereichs
- F_VN_CopyImageRegionToRegion [▶ 802] zum Kopieren eines Bildbereichs an eine bestimmte Position

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.5 F_VN_CopyImageRegion

Deep copy the specified region of interest into a new image.

Syntax

Definition:

```
FUNCTION F_VN_CopyImageRegion : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    nX          : UDINT;
    nY          : UDINT;
    nWidth      : UDINT;
    nHeight     : UDINT;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
nX	UDINT	Left boundary (inclusive 0-based index)
nY	UDINT	Upper boundary (inclusive 0-based index)
nWidth	UDINT	ROI width
nHeight	UDINT	ROI height
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CopyImageRegion` erstellt eine tiefe Kopie eines Bildausschnitts des Eingangsbilds `ipSrcImage`. Der Bildausschnitt wird wie bei einer ROI [▶ 168] durch ein Rechteck mit Position und Größe angegeben.

Rechenzeitoptimierung

Prüfen Sie in Bezug auf Rechenzeitoptimierung jede Bildkopie auf ihre Notwendigkeit.

Wenn keine Kopie benötigt wird, sondern nur der Bildausschnitt verkleinert werden soll, kann stattdessen eine Region of Interest [▶ 168] gesetzt werden.

Anwendung

Das Kopieren eines Bildbereichs der Größe `[240, 120]` mit oberer linker Ecke an Position `[50, 50]` sieht z. B. so aus:

```
hr := F_VN_CopyImageRegion(
    ipSrcImage  :=  ipImageIn,
    nX          :=  50,
    nY          :=  50,
    nWidth      :=  240,
    nHeight     :=  120,
    ipDestImage :=  ipImageWork,
    hrPrev      :=  hr,
);
```

Verwandte Funktionen

- F_VN_CopyImage [▶ 799] zum Kopieren des gesamten Bildes
- F_VN_CopyImageRegion [▶ 801] zum Kopieren eines Bildbereichs
- F_VN_CopyImageRegionToRegion [▶ 802] zum Kopieren eines Bildbereichs an eine bestimmte Position

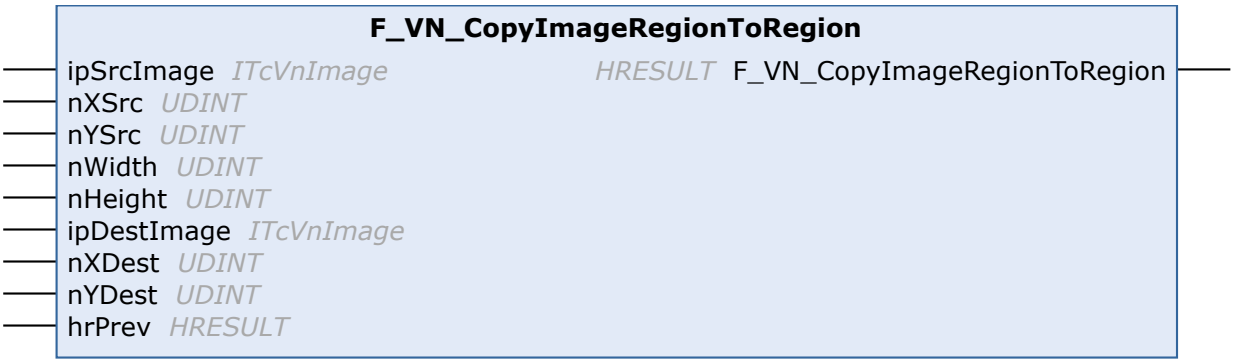
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.6 F_VN_CopyImageRegionToRegion



Copy an image region into another image region.

Syntax

Definition:

```
FUNCTION F_VN_CopyImageRegionToRegion : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    nXSrc       : UDINT;
    nYSrc       : UDINT;
    nWidth      : UDINT;
    nHeight     : UDINT;
    ipDestImage : ITcVnImage;
    nXDest      : UDINT;
    nYDest      : UDINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
nXSrc	UDINT	Left boundary in ipSrcImage (inclusive 0-based index)
nYSrc	UDINT	Upper boundary in ipSrcImage (inclusive 0-based index)
nWidth	UDINT	ROI width
nHeight	UDINT	ROI height
ipDestImage	ITcVnImage [▶ 426]	Destination image (same type as ipSrcImage)
nXDest	UDINT	Left boundary in ipDestImage (inclusive 0-based index)
nYDest	UDINT	Upper boundary in ipDestImage (inclusive 0-based index)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CopyImageRegionToRegion` kopiert einen Bildausschnitt des Eingangsbildes `ipSrcImage` an eine Stelle im Zielbild `ipDestImage`. Der Bildausschnitt wird wie bei einer ROI [▶ 168] durch ein Rechteck mit Position und Größe angegeben.

Anwendung

Das Kopieren eines `[100, 100]`-großen Bildbereichs an der Position `aPosition` in die linke obere Ecke des Ergebnisbildes `ipImageRes` sieht z. B. so aus:

```
hr := F_VN_CopyImageRegionToRegion(
    ipSrcImage  :=  ipImageWork,
    nXSrc       :=  aPosition[0],
    nYSrc       :=  aPosition[1],
    nWidth      :=  100,
    nHeight     :=  100,
    ipDestImage :=  ipImageRes,
    nXDest      :=  0,
    nYDest      :=  0,
    hrPrev      :=  hr
);
```

Beispiele

- Bildbereiche kopieren [▶ 2843]

Verwandte Funktionen

- F_VN_CopyImage [▶ 799] zum Kopieren des gesamten Bildes
- F_VN_CopyImageRegion [▶ 801] zum Kopieren eines Bildbereichs
- F_VN_CopyImageRegionToRegion [▶ 802] zum Kopieren eines Bildbereichs an eine bestimmte Position

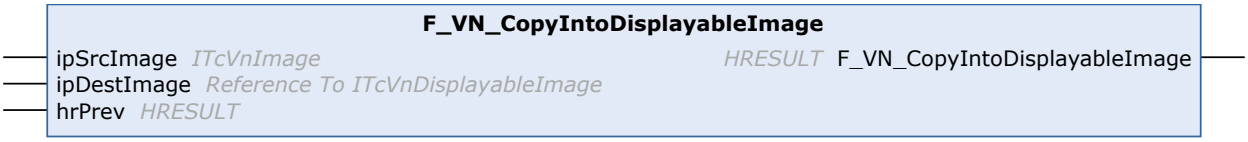
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.7 F_VN_CopyIntoDisplayableImage



Copys an image into a displayable image. If you do not want to use ipSrcImage after this function call, you might want to use F_VN_TransformIntoDisplayableImage instead for better performance.

Syntax

Definition:

```
FUNCTION F_VN_CopyIntoDisplayableImage : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnDisplayableImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnDisplayableImage [▶ 426]	Returns the displayable image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Für weitere Informationen siehe Anzeigbare Bilder [▶ 161].

Verwandte Funktionen

- F_VN_TransformIntoDisplayableImage [▶ 835]
- F_VN_TransformIntoDisplayableImageExp [▶ 836]

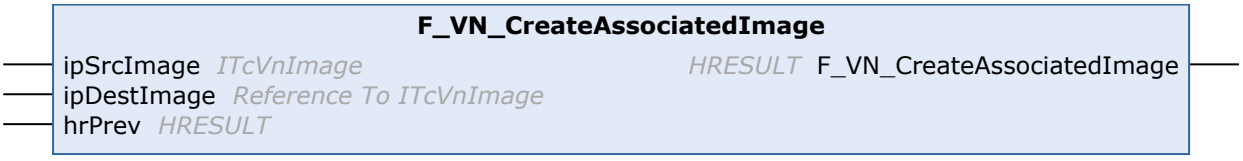
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.8 F_VN_CreateAssociatedImage



Creates a new image that shares its data with the source image. E.g. useful to work on different (disjoint) ROIs in parallel.

Syntax

Definition:

```
FUNCTION F_VN_CreateAssociatedImage : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Returns the created image (Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.9 F_VN_CreateEmptyImage

Creates an empty image without allocating any data buffer. The initial reference count is set to one if a new image interface is created and kept, otherwise.

Syntax

Definition:

```
FUNCTION F_VN_CreateEmptyImage : HRESULT
VAR_INPUT
    ipImage : Reference To ITcVnImage;
    hrPrev  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	Reference To ITcVnImage [▶ 426]	Returns the created image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

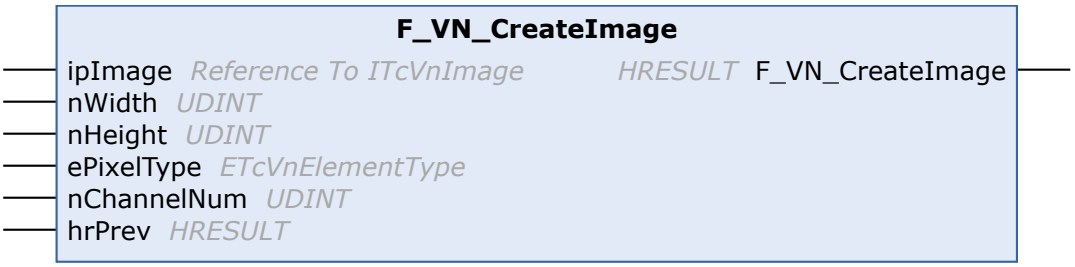
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.10 F_VN_CreateImage



Creates an image and allocate an appropriate data buffer. The initial reference count is set to one if a new image interface is created and kept, otherwise.

Syntax

Definition:

```
FUNCTION F_VN_CreateImage : HRESULT
VAR_INPUT
    ipImage     : Reference To ITcVnImage;
    nWidth      : UDINT;
    nHeight     : UDINT;
    ePixelType  : ETcVnElementType;
    nChannelNum : UDINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	Reference To ITcVnImage [▶ 426]	Returns the created image (Non-zero interface pointers are reused.)
nWidth	UDINT	Image width
nHeight	UDINT	Image height
ePixelType	ETcVnElementType [▶ 210]	Pixel type
nChannelNum	UDINT	Number of channels (1 to 255)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

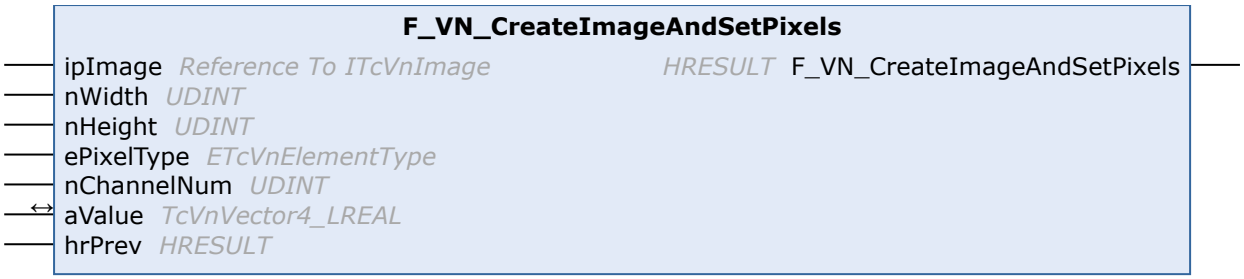
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.11 F_VN_CreateImageAndSetPixels



Creates an image, allocates an appropriate data buffer and sets all pixels to the specified value. The initial reference count is set to one if a new image interface is created and kept, otherwise.

Syntax

Definition:

```
FUNCTION F_VN_CreateImageAndSetPixels : HRESULT
VAR_INPUT
    ipImage     : Reference To ITcVnImage;
    nWidth      : UDINT;
    nHeight     : UDINT;
    ePixelType  : ETcVnElementType;
    nChannelNum : UDINT;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	Reference To ITcVnImage [▶ 426]	Returns the created image (Non-zero interface pointers are reused.)
nWidth	UDINT	Image width
nHeight	UDINT	Image height
ePixelType	ETcVnElementType [▶ 210]	Pixel type
nChannelNum	UDINT	Number of channels (1 to 4)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_LREAL [▶ 173]	Pixel value

Return value

HRESULT [▶ 150]

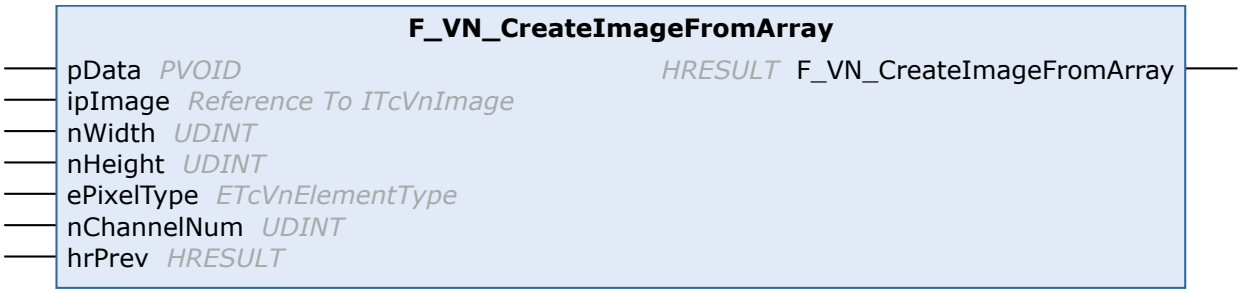
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.12 F_VN_CreateImageFromArray



Creates an image and initialize it with the provided data. The initial reference count is set to one if a new image interface is created and kept, otherwise.

Syntax

Definition:

```
FUNCTION F_VN_CreateImageFromArray : HRESULT
VAR_INPUT
    pData       : PVOID;
    ipImage     : Reference To ITcVnImage;
    nWidth      : UDINT;
    nHeight     : UDINT;
    ePixelType  : ETcVnElementType;
```

```
    nChannelNum : UDINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
pData	PVOID	Pointer to the 1D data array. Make sure that the array contains at least nWidth * nHeight * nChannelNum elements and that the array element type matches ePixelType.
ipImage	Reference To ITcVnImage [▶ 426]	Returns the created image (Non-zero interface pointers are reused.)
nWidth	UDINT	Image width
nHeight	UDINT	Image height
ePixelType	ETcVnElementType [▶ 210]	Pixel type
nChannelNum	UDINT	Number of channels (1 to 255)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.13 F_VN_ExportImage

Exports the image data into a given buffer (e.g. an array). F_VN_ExportImageSize should be called before to get the required buffer size.

Syntax

Definition:

```
FUNCTION F_VN_ExportImage : HRESULT
VAR_INPUT
    ipImage     : ITcVnImage;
    pBuffer     : PVOID;
    nBufferSize : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Image
pBuffer	PVOID	Pointer to a buffer to store the image data (Make sure to allocate enough memory! The required size in bytes can be determined using the function F_VN_ExportImageSize)
nBufferSize	ULINT	Size of the buffer memory in bytes
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.14 F_VN_ExportImageAsBmp

Exports the image as bitmap into a given buffer (e.g. an array). F_VN_ExportImageAsBmpSize should be called before to get the required buffer size.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_ExportImageAsBmp : HRESULT
VAR_INPUT
    ipImage     : ITcVnImage;
    pBuffer     : PVOID;
    nBufferSize : ULINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Image
pBuffer	PVOID	Pointer to a buffer to store the bitmap image (Make sure to allocate enough memory! The required size in bytes can be determined using the function F_VN_ExportImageAsBmpSize)
nBufferSize	ULINT	Size of the buffer memory in bytes
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Beachten Sie die Konvertierungseigenschaften der Schnittstelle ITcVnBitmapExport [▶ 409].

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.15 F_VN_ExportImageAsBmpExp

Exports the image as bitmap into a given buffer (e.g. an array). F_VN_ExportImageAsBmpSizeExp should be called before to get the required buffer size. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_ExportImageAsBmpExp : HRESULT
VAR_INPUT
    ipImage     : ITcVnImage;
    pBuffer     : PVOID;
    nBufferSize : ULINT;
    nWidth      : UDINT;
    nHeight     : UDINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Image
pBuffer	PVOID	Pointer to a buffer to store the bitmap image (Make sure to allocate enough memory! The required size in bytes can be determined using the function F_VN_ExportImageAsBmpSizeExp)
nBufferSize	ULINT	Size of the buffer memory in bytes
nWidth	UDINT	Desired width (or 0 to keep the original width)
nHeight	UDINT	Desired height (or 0 to keep the original height)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.16 F_VN_ExportImageAsBmpSize

Get the required buffer size (in bytes) to store the image as bitmap.

Syntax

Definition:

```
FUNCTION F_VN_ExportImageAsBmpSize : HRESULT
VAR_INPUT
    ipImage     : ITcVnImage;
END_VAR
VAR_IN_OUT
    nBufferSize : ULINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nBufferSize	ULINT	Returns the required buffer size in bytes

Return value

HRESULT [▶ 150]

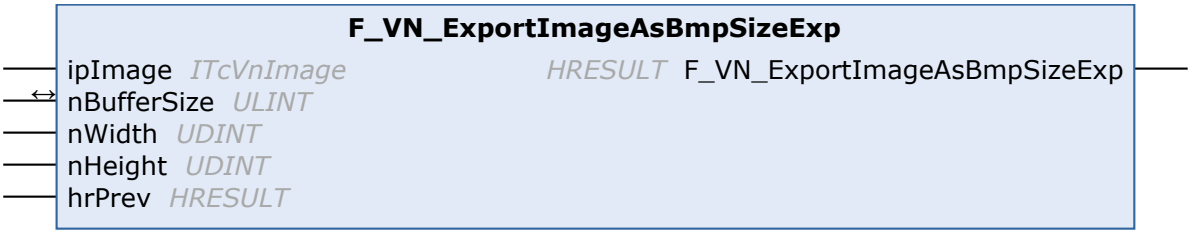
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.17 F_VN_ExportImageAsBmpSizeExp



Get the required buffer size (in bytes) to store the image as bitmap. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_ExportImageAsBmpSizeExp : HRESULT
VAR_INPUT
    ipImage     : ITcVnImage;
END_VAR
VAR_IN_OUT
    nBufferSize : ULINT;
END_VAR
VAR_INPUT
    nWidth      : UDINT;
    nHeight     : UDINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Image
nWidth	UDINT	Desired width (or 0 to keep the original width)
nHeight	UDINT	Desired height (or 0 to keep the original height)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nBufferSize	ULINT	Returns the required buffer size in bytes

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.18 F_VN_ExportImageSize

Get the required buffer size (in bytes) to store the image data.

Syntax

Definition:

```
FUNCTION F_VN_ExportImageSize : HRESULT
VAR_INPUT
    ipImage     : ITcVnImage;
END_VAR
VAR_IN_OUT
    nBufferSize : ULINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nBufferSize	ULINT	Returns the required buffer size in bytes

Return value

HRESULT [▶ 150]

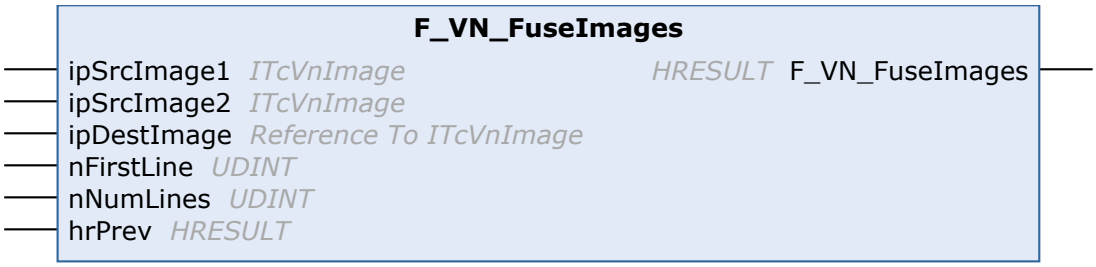
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.19 F_VN_FuseImages



Fuse 2 images vertically (intended for line scan cameras).

Syntax

Definition:

```
FUNCTION F_VN_FuseImages : HRESULT
VAR_INPUT
    ipSrcImage1 : ITcVnImage;
    ipSrcImage2 : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    nFirstLine  : UDINT;
    nNumLines   : UDINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage1	ITcVnImage [▶ 426]	Source image 1
ipSrcImage2	ITcVnImage [▶ 426]	Source image 2
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image
nFirstLine	UDINT	Line index in ipSrcImage1, which is the first line in ipDestImage
nNumLines	UDINT	The number of lines that should be copied to ipDestImage, starting with nFirstLine in ipSrcImage1. Once the last line of ipSrcImage1 was copied, the remaining lines are copied from ipSrcImage2, starting at line index 0.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

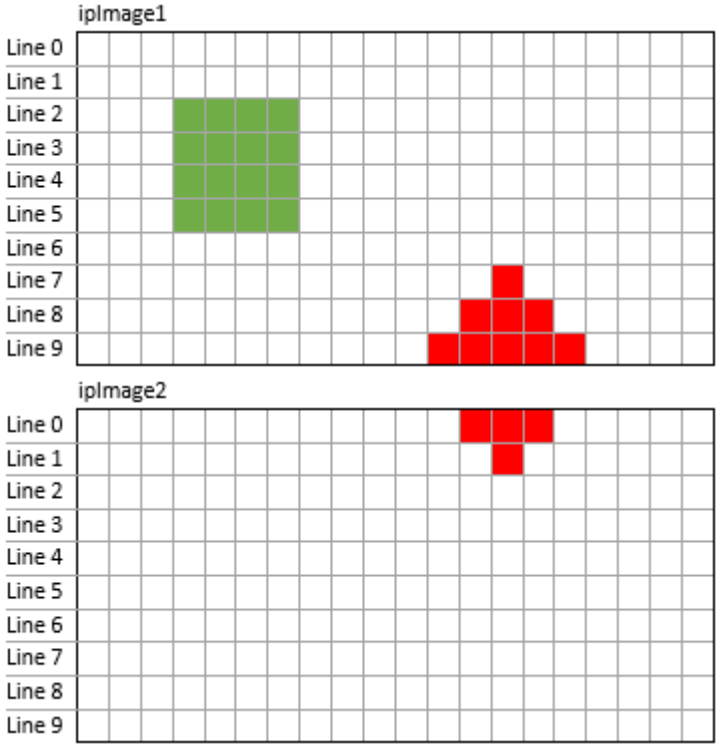
Weiterführende Informationen

Die Funktion `F_VN_FuseImages` verknüpft zwei Bilder untereinander, sodass ein neues, zusammenhängendes Bild entsteht. Dies ist insbesondere bei Aufnahmen von Zeilenkameras interessant.

Für die Verknüpfung von bis zu 10 Bildern kann die Funktion F_VN_FuseImagesArray [▶ 817] genutzt werden.

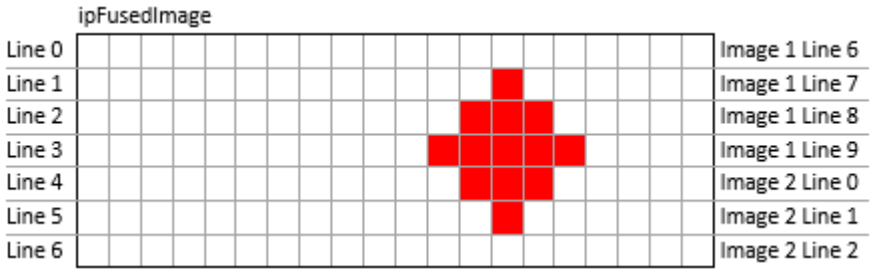
Anwendung

Zeilenkameras geben meist eine vorgegebene Anzahl nacheinander aufgenommener Zeilen als 2D-Bild zurück. Dieses 2D-Bild kann ein interessantes Objekt (im nachfolgenden Bild grün dargestellt) vollständig enthalten, es kann aber auch nur ein Objektteil (rot dargestellt) im Bild enthalten sein.



Zur vollständigen Analyse des roten Objektes können mit der Funktion `F_VN_FuseImages` die beiden Bilder mit den Objektteilen zu einem Bild verknüpft werden. Ist bereits bekannt, welche Zeilen das Objekt betreffen, kann die Verknüpfung auch auf die relevanten Zeilen begrenzt werden.

```
F_VN_FuseImages(ipImage1, ipImage2, ipFusedImage, 6, 7, S_OK);
```



Beispiele

- Mehrere Bilder zusammenfügen [▶ 2842]

Verwandte Funktionen

- F_VN_FuseImages [▶ 815] zur Verknüpfung von zwei Bildern
- F_VN_FuseImagesArray [▶ 817] zur Verknüpfung von bis zu 10 Bildern

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.20 F_VN_FuseImagesArray

Fuse up to 10 images vertically (intended for line scan cameras).

Syntax

Definition:

```
FUNCTION F_VN_FuseImagesArray : HRESULT
VAR_IN_OUT
    aSrcImageArray : TcVnArray10_ITcVnImage;
END_VAR
VAR_INPUT
    nValidImages   : UDINT;
    ipDestImage    : Reference To ITcVnImage;
    nFirstLine     : UDINT;
    nNumLines      : UDINT;
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nValidImages	UDINT	Number of valid images in aSrcImageArray
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image
nFirstLine	UDINT	Line index in ipSrcImage1, which is the first line in ipDestImage
nNumLines	UDINT	The number of lines that should be copied to ipDestImage, starting with nFirstLine in aSrcImageArray[0]. Once the last line of aSrcImageArray[0] was copied, the remaining lines are copied from aSrcImageArray[1] (starting at line index 0) and so on.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aSrcImageArray	TcVnArray10_ITcVnImage [▶ 173]	Ten-element source image array (not all elements need to be filled, nValidImages specifies the actual amount of images)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_FuseImageArray` ist eine Abwandlung der Funktion F_VN_FuseImages [▶ 815] und kann statt zwei Bildern bis zu zehn Bilder aneinanderfügen.

Beispiele

- Mehrere Bilder zusammenfügen [▶ 2842]

Verwandte Funktionen

- F_VN_FuseImages [▶ 815] zur Verknüpfung von zwei Bildern
- F_VN_FuseImagesArray [▶ 817] zur Verknüpfung von bis zu 10 Bildern

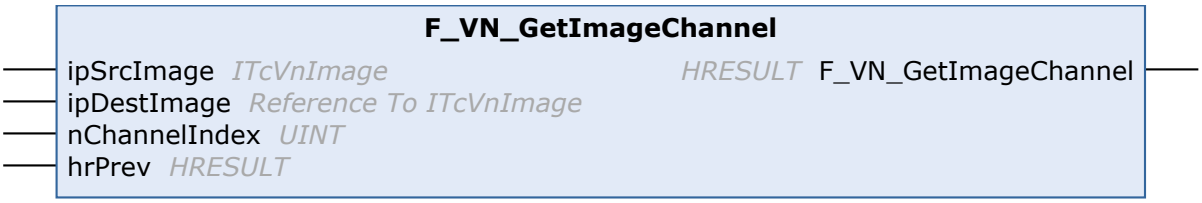
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.21 F_VN_GetImageChannel



Return the specified source image channel as a single-channel image.

Syntax

Definition:

```
FUNCTION F_VN_GetImageChannel : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipDestImage   : Reference To ITcVnImage;
    nChannelIndex : UINT;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate image will be created if required.)
nChannelIndex	UINT	Index of the requested source image channel
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.22 F_VN_GetImageHeight

Return the height of an image. (Alternatively use interface method .GetHeight.)

Syntax

Definition:

```
FUNCTION F_VN_GetImageHeight : HRESULT
VAR_INPUT
    ipImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    nHeight : UDINT;
END_VAR
VAR_INPUT
    hrPrev  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nHeight	UDINT	Return the height of ipImage

Return value

HRESULT [▶ 150]

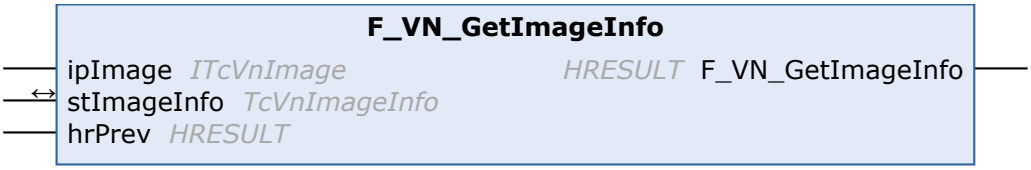
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.23 F_VN_GetImageInfo



Gets a struct containing all common meta infos of the image. (Alternatively use interface method .GetImageInfo.)

Syntax

Definition:

```
FUNCTION F_VN_GetImageInfo : HRESULT
VAR_INPUT
    ipImage     : ITcVnImage;
END_VAR
VAR_IN_OUT
    stImageInfo : TcVnImageInfo;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stImageInfo	TcVnImageInfo [▶ 244]	Returns a struct describing the image

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.24 F_VN_GetImageWidth

Return the width of an image. (Alternatively use interface method .GetWidth.)

Syntax

Definition:

```
FUNCTION F_VN_GetImageWidth : HRESULT
VAR_INPUT
    ipImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    nWidth  : UDINT;
END_VAR
VAR_INPUT
    hrPrev  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nWidth	UDINT	Return the width of ipImage

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.25 F_VN_GetPixel

Gets a specific pixel.

Syntax

Definition:

```
FUNCTION F_VN_GetPixel : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aValue     : TcVnVector4_LREAL;
```

```
END_VAR
VAR_INPUT
    nX          : UDINT;
    nY          : UDINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
nX	UDINT	x coordinate of the pixel
nY	UDINT	y coordinate of the pixel
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_LREAL [▶ 173]	Returns the pixel value (Unused channels are set to 0.)

Return value

HRESULT [▶ 150]

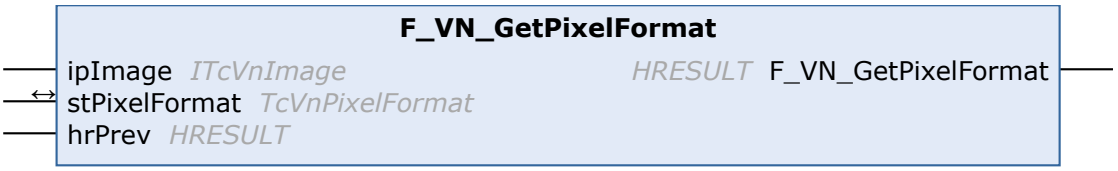
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.26 F_VN_GetPixelFormat



Gets the pixel format of an image. (Alternatively use interface method .GetPixelFormat.)

Syntax

Definition:

```
FUNCTION F_VN_GetPixelFormat : HRESULT
VAR_INPUT
    ipImage       : ITcVnImage;
END_VAR
VAR_IN_OUT
    stPixelFormat : TcVnPixelFormat;
END_VAR
VAR_INPUT
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stPixelFormat	TcVnPixelFormat [▶ 256]	Returns a struct describing the pixel format

Return value

HRESULT [▶ 150]

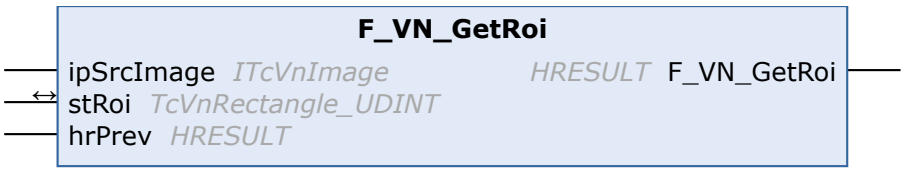
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.27 F_VN_GetRoi



Gets the coordinates of the region of interest (ROI) within the image.

Syntax

Definition:

```
FUNCTION F_VN_GetRoi : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    stRoi      : TcVnRectangle_UDINT;
END_VAR
VAR_INPUT
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRoi	TcVnRectangle_UDINT [▶ 258]	Returns the coordinates of the region of interest

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_GetRoi` zeigt an, welche Region Of Interest [▶ 168] zurzeit auf dem Eingangsbild `ipSrcImage` gesetzt ist. Die Information wird z. B. benötigt, wenn Punkte in dem Bild in Relation zum Gesamtbild angegeben werden sollen oder wenn die ROI dynamisch vergrößert werden soll.

Anwendung

Das Beziehen eines Punktes auf die Koordinaten des Originalbildes sieht z. B. so aus:

```
hr := F_VN_ContourCenterOfMass(ipContour, aCenter, hr);
hr := F_VN_GetRoi(ipImageWork, stRoi, hr);
aCenter[0] := aCenter[0] + stRoi.nX;
aCenter[1] := aCenter[1] + stRoi.nY;
```

Verwandte Funktionen

- F_VN_SetRoi [▶ 830] zum Setzen einer Region of Interest
- F_VN_GetRoi [▶ 823] zum Abrufen der gesetzten Region of Interest
- F_VN_ResetRoi [▶ 825] zum Zurücksetzen der Region of Interest auf das gesamte Bild
- F_VN_CopyImageRegion [▶ 801] zum Kopieren eines Bildausschnitts

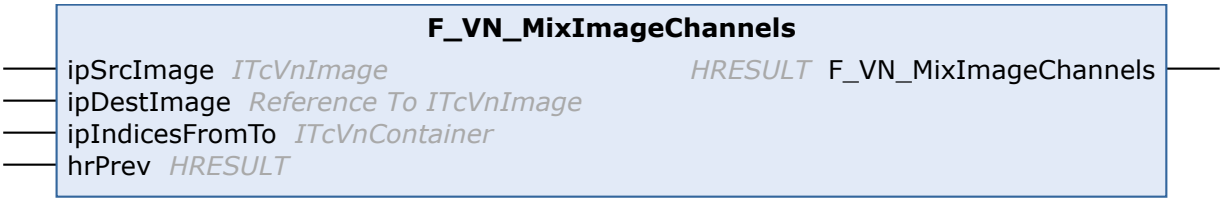
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.28 F_VN_MixImageChannels



Mix image channels by copying the specified channels of the source image into the specified channels of the destination image.

Syntax

Definition:

```
FUNCTION F_VN_MixImageChannels : HRESULT
VAR_INPUT
    ipSrcImage      : ITcVnImage;
```

```
    ipDestImage     : Reference To ITcVnImage;
    ipIndicesFromTo : ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate image will be created if required.)
ipIndicesFromTo	ITcVnContainer [▶ 385]	Index pairs (ContainerType_Vector_TcVnVector2_DINT), specifying which source channel (TcVnVector2_DINT [0]) should be copied to which destination channel (TcVnVector2_DINT [1]).
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

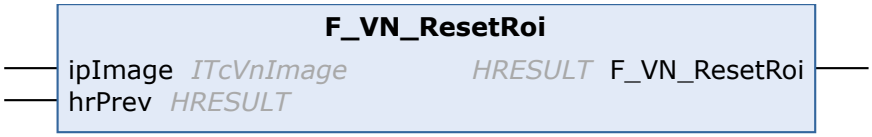
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.29 F_VN_ResetRoi



Reset the region of interest (ROI) of an image. (After this operation, the ROI is set to the entire image.)

Syntax

Definition:

```
FUNCTION F_VN_ResetRoi : HRESULT
VAR_INPUT
    ipImage : ITcVnImage;
    hrPrev  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_ResetRoi` setzt die Region of Interest [▶ 168] eines Bildes zurück. Nach Ausführung der Funktion steht die ROI des Bilds auf Position `[0, 0]` und Größe `[ImageWidth, ImageHeight]`.

Anwendung

Das Zurücksetzen einer ROI sieht z. B. so aus:

```
Hr:= F_VN_ResetRoi(ipImageWork, hr);
```

Verwandte Funktionen

- F_VN_SetRoi [▶ 830] zum Setzen einer Region of Interest
- F_VN_GetRoi [▶ 823] zum Abrufen der gesetzten Region of Interest
- F_VN_ResetRoi [▶ 825] zum Zurücksetzen der Region of Interest auf das gesamte Bild
- F_VN_CopyImageRegion [▶ 801] zum Kopieren eines Bildausschnitts

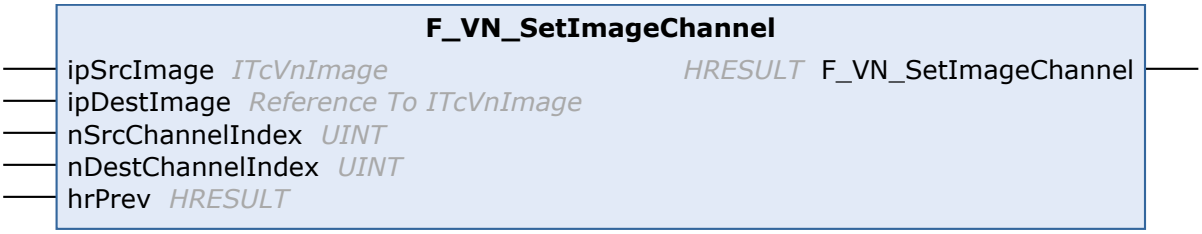
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.30 F_VN_SetImageChannel



Sets the specified destination image channel to the values of the specified source image channel.

Syntax

Definition:

```
FUNCTION F_VN_SetImageChannel : HRESULT
VAR_INPUT
    ipSrcImage        : ITcVnImage;
    ipDestImage       : Reference To ITcVnImage;
    nSrcChannelIndex  : UINT;
    nDestChannelIndex : UINT;
    hrPrev            : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate image will be created if required.)
nSrcChannelIndex	UINT	Index of the source image channel
nDestChannelIndex	UINT	Index of the destination image channel, which will be replaced by the specified source image channel
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

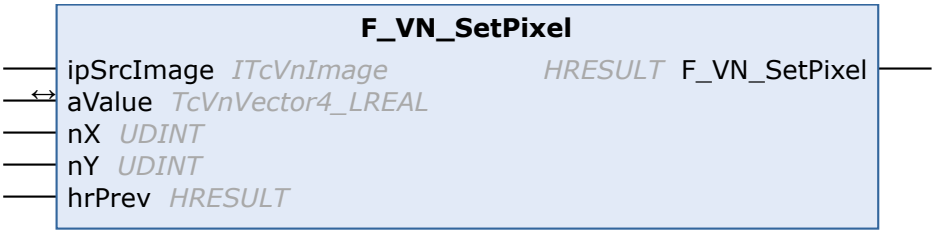
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.31 F_VN_SetPixel



Sets a specific pixel.

Syntax

Definition:

```
FUNCTION F_VN_SetPixel : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aValue     : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nX         : UDINT;
    nY         : UDINT;
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
nX	UDINT	x coordinate of the pixel
nY	UDINT	y coordinate of the pixel
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_LREAL [▶ 173]	The pixel value to set (additional channels are ignored.)

Return value

HRESULT [▶ 150]

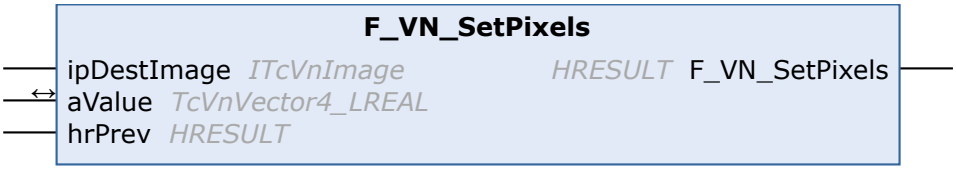
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.32 F_VN_SetPixels



Sets all pixels of an image to a given value.

Syntax

Definition:

```
FUNCTION F_VN_SetPixels : HRESULT
VAR_INPUT
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipDestImage	ITcVnImage [▶ 426]	Destination image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_LREAL [▶ 173]	Pixel value

Return value

HRESULT [▶ 150]

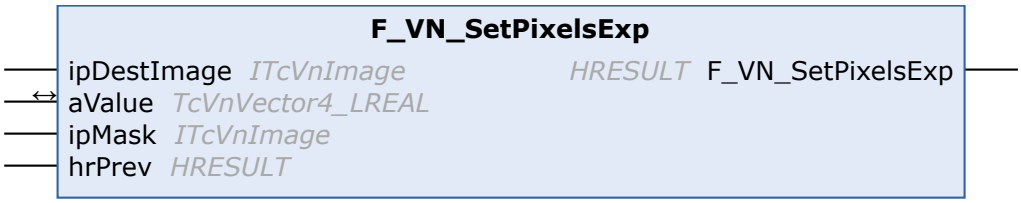
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.33 F_VN_SetPixelsExp



Sets all pixels of an image to a given value. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_SetPixelsExp : HRESULT
VAR_INPUT
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aValue      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipMask      : ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipDestImage	ITcVnImage [▶ 426]	Destination image
ipMask	ITcVnImage [▶ 426]	Mask image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aValue	TcVnVector4_LREAL [▶ 173]	Pixel value

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.34 F_VN_SetRoi

Sets a region of interest (ROI) within an image.

Syntax

Definition:

```
FUNCTION F_VN_SetRoi : HRESULT
VAR_INPUT
    nX          : UDINT;
    nY          : UDINT;
    nWidth      : UDINT;
    nHeight     : UDINT;
    ipDestImage : ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nX	UDINT	Left boundary (inclusive 0-based index)
nY	UDINT	Upper boundary (inclusive 0-based index)
nWidth	UDINT	ROI width
nHeight	UDINT	ROI height
ipDestImage	ITcVnImage [▶ 426]	Destination image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_SetRoi` setzt die Region of Interest [▶ 168] eines Bildes. Falls die Rechteck-Parameter nicht einzeln, sondern als Struktur vom Typ TcVnRectangle_UDINT [▶ 258] übergeben werden soll, kann die Funktion F_VN_SetRoi_TcVnRectangle_UDINT [▶ 832] genutzt werden.

Anwendung

Das Setzen einer ROI mit der Größe `[240, 120]` an der Position `[50, 50]` sieht z. B. so aus:

```
hr := F_VN_SetRoi(50, 50, 240, 120, ipImageWork, hr);
```

Verwandte Funktionen

- F_VN_SetRoi [▶ 830] zum Setzen einer Region of Interest
- F_VN_GetRoi [▶ 823] zum Abrufen der gesetzten Region of Interest
- F_VN_ResetRoi [▶ 825] zum Zurücksetzen der Region of Interest auf das gesamte Bild
- F_VN_CopyImageRegion [▶ 801] zum Kopieren eines Bildausschnitts

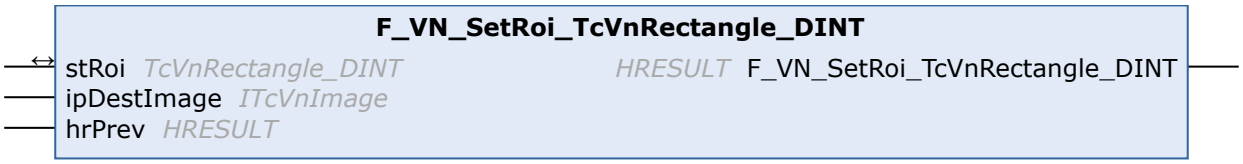
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.35 F_VN_SetRoi_TcVnRectangle_DINT



Sets a region of interest (ROI) within an image.

Syntax

Definition:

```
FUNCTION F_VN_SetRoi_TcVnRectangle_DINT : HRESULT
VAR_IN_OUT
    stRoi       : TcVnRectangle_DINT;
END_VAR
VAR_INPUT
    ipDestImage : ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipDestImage	ITcVnImage [▶ 426]	Destination image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRoi	TcVnRectangle_DINT [▶ 257]	Region of interest

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.36 F_VN_SetRoi_TcVnRectangle_UDINT

Sets a region of interest (ROI) within an image.

Syntax

Definition:

```
FUNCTION F_VN_SetRoi_TcVnRectangle_UDINT : HRESULT
VAR_IN_OUT
    stRoi       : TcVnRectangle_UDINT;
END_VAR
VAR_INPUT
    ipDestImage : ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipDestImage	ITcVnImage [▶ 426]	Destination image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRoi	TcVnRectangle_UDINT [▶ 258]	Region of interest

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_SetRoi_TcVnRectangle_UDINT` setzt wie die Funktion F_VN_SetRoi [▶ 830], eine Region of Interest [▶ 168]. Sie nimmt die Rechteck-Parameter allerdings nicht einzeln, sondern als Struktur vom Typ TcVnRectangle_UDINT [▶ 258] entgegen.

Dies ermöglicht es zum Beispiel, mit dem Ergebnis der Funktion F_VN_UprightBoundingRectangle [▶ 1018] direkt die ROI zu setzen.

Anwendung

Das Setzen einer ROI um eine gefundene Kontur [▶ 165] sieht z. B.: so aus:

```
hr := F_VN_UprightBoundingRectangle(ipContour, stRoi, hr);
hr := F_VN_SetRoi_TcVnRectangle_UDINT(stRoi, ipImageWork, hr);
```

Verwandte Funktionen

- F_VN_SetRoi [▶ 830] zum Setzen einer Region of Interest
- F_VN_GetRoi [▶ 823] zum Abrufen der gesetzten Region of Interest
- F_VN_ResetRoi [▶ 825] zum Zurücksetzen der Region of Interest auf das gesamte Bild
- F_VN_CopyImageRegion [▶ 801] zum Kopieren eines Bildausschnitts

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.37 F_VN_SplitImageChannels

Split a multi-channel image into multiple single channel images (1 for each source image channel).

Syntax

Definition:

```
FUNCTION F_VN_SplitImageChannels : HRESULT
VAR_INPUT
    ipSrcImage     : ITcVnImage;
    pDestImages    : PVOID;
    nArraySize     : UINT;
    hrPrev         : HRESULT;
END_VAR
VAR_OUTPUT
    nImageChannels : UINT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image with multiple channels
pDestImages	PVOID	Pointer to an array of ITcVnImage (appropriate destination images will be created if required)
nArraySize	UINT	Number of pDestImages array elements (array must be >= ipSourceImage channels)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Outputs

Name	Type	Description
nImageChannels	UINT	Actual source image channels

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.38 F_VN_TransformIntoDisplayableImage

Transform an image into a displayable image. The source image will be released and zeroed while existing destination images will be released and overwritten with the function result. The source image must not be used anywhere else and the transformation is very fast. If you want to use ipSrcImage after this function call, use F_VN_CopyIntoDisplayableImage instead.

Syntax

Definition:

```
FUNCTION F_VN_TransformIntoDisplayableImage : HRESULT
VAR_INPUT
    ipSrcImage  : Reference To ITcVnImage;
    ipDestImage : Reference To ITcVnDisplayableImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	Reference To ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnDisplayableImage [▶ 426]	Returns the displayable image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Für weitere Informationen siehe Anzeigbare Bilder [▶ 161].

HINWEIS

Verwendung mit Ads Communicator Objekt

Beachten Sie den Fall, wenn das zu transformierende Bild aus einem Kameraobjekt stammt, dessen Ads Communicator Objekt aktiv ist! Dann ist es möglich, dass das Bild intern in dem Objekt noch nicht freigegeben ist und daher nicht in ein anzeigbares Bild transformiert werden kann. Nutzen Sie in dem Fall F_VN_CopyIntoDisplayableImage [▶ 804] oder F_VN_TransformIntoDisplayableImageExp [▶ 836] mit `bAllowDeepCopy := TRUE`.

Verwandte Funktionen

- F_VN_CopyIntoDisplayableImage [▶ 804]
- F_VN_TransformIntoDisplayableImageExp [▶ 836]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.39 F_VN_TransformIntoDisplayableImageExp

Transform an image into a displayable image. The source image will be released and zeroed while existing destination images will be released and overwritten with the function result. If bAllowDeepCopy equals false, the source image must not be used anywhere else and the transformation is always very fast. Otherwise, if bAllowDeepCopy equals true, a deep copy of the source image might be created if required, which will result in a longer execution time. If you want to use ipSrcImage after this function call, use F_VN_CopyIntoDisplayableImage instead. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_TransformIntoDisplayableImageExp : HRESULT
VAR_INPUT
    ipSrcImage     : Reference To ITcVnImage;
    ipDestImage    : Reference To ITcVnDisplayableImage;
    bAllowDeepCopy : BOOL;
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	Reference To ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnDisplayableImage [▶ 426]	Returns the displayable image
bAllowDeepCopy	BOOL	Allow deep image copies, if required
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Für weitere Informationen siehe Anzeigbare Bilder [▶ 161].

Parameter

Erlaubnis für tiefe Kopie

Mit dem Parameter `bAllowDeepCopy` kann die Erlaubnis (`TRUE`) für eine tiefe Kopie gegeben werden. Dies ist relevant, wenn das zur transformierende Bild von einem Kameraobjekt stammt, dessen Ads Communicator Objekt [▶ 133] aktiviert ist und das Bild daher intern referenziert sein kann.

Falls keine Erlaubnis gegeben werden soll (`FALSE`) kann auch die Standard-Variante F_VN_TransformIntoDisplayableImage [▶ 835] verwendet werden.

Verwandte Funktionen

- F_VN_CopyIntoDisplayableImage [▶ 804]
- F_VN_TransformIntoDisplayableImage [▶ 835]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.40 F_VN_CreateNDimensionalImage

Creates an N-dimensional image and allocates an appropriate data buffer. The initial reference count is set to one if a new image interface is created and kept, otherwise.

Syntax

Definition:

```
FUNCTION F_VN_CreateNDimensionalImage : HRESULT
VAR_INPUT
    ipImage    : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    aShape     : TcVnArray33_UDINT;
END_VAR
VAR_INPUT
    ePixelType : ETcVnElementType;
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	Reference To ITcVnImage [▶ 426]	Returns the created image (Non-zero interface pointers are reused.)
ePixelType	ETcVnElementType [▶ 210]	Pixel type
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aShape	TcVnArray33_UDINT [▶ 173]	Specifies the shape of ipDestImage. First entry is the number of dimensions of ipImage (max. 32), followed by the sizes of each dimension.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.41 F_VN_CreateNDimensionalImageAndSetPixels

Creates an N-dimensional image, allocates an appropriate data buffer and sets all pixels to the specified value. The initial reference count is set to one if a new image interface is created and kept, otherwise.

Syntax

Definition:

```
FUNCTION F_VN_CreateNDimensionalImageAndSetPixels : HRESULT
VAR_INPUT
    ipImage    : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    aShape     : TcVnArray33_UDINT;
END_VAR
VAR_INPUT
    ePixelType : ETcVnElementType;
    fValue     : LREAL;
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	Reference To ITcVnImage [▶ 426]	Returns the created image (Non-zero interface pointers are reused.)
ePixelType	ETcVnElementType [▶ 210]	Pixel type
fValue	LREAL	Pixel value
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aShape	TcVnArray33_UDINT [▶ 173]	Specifies the shape of ipDestImage. First entry is the number of dimensions of ipImage (max. 32), followed by the sizes of each dimension.

Return value

HRESULT [▶ 150]

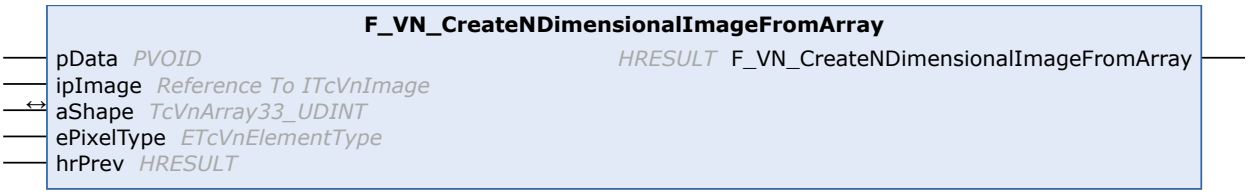
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.42 F_VN_CreateNDimensionalImageFromArray



Creates an N-dimensional image and initialize it with the provided data. The initial reference count is set to one if a new image interface is created and kept, otherwise.

Syntax

Definition:

```
FUNCTION F_VN_CreateNDimensionalImageFromArray : HRESULT
VAR_INPUT
    pData      : PVOID;
    ipImage    : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    aShape     : TcVnArray33_UDINT;
END_VAR
VAR_INPUT
    ePixelType : ETcVnElementType;
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
pData	PVOID	Pointer to the 1D data array. Ensure that the array contains at least the number of elements equal to the product of all dimensions in aShape and that the array element type matches ePixelType.
ipImage	Reference To ITcVnImage [▶ 426]	Returns the created image (Non-zero interface pointers are reused.)
ePixelType	ETcVnElementType [▶ 210]	Pixel type
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aShape	TcVnArray33_UDINT [▶ 173]	Specifies the shape of ipDestImage. First entry is the number of dimensions of ipImage (max. 32), followed by the sizes of each dimension.

Return value

HRESULT [▶ 150]

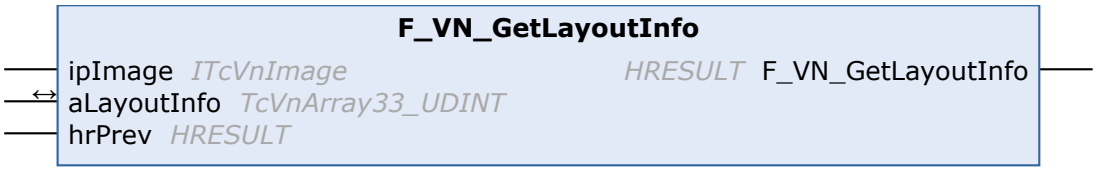
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.43 F_VN_GetLayoutInfo



Return the layout information of an image

Syntax

Definition:

```
FUNCTION F_VN_GetLayoutInfo : HRESULT
VAR_INPUT
    ipImage     : ITcVnImage;
END_VAR
VAR_IN_OUT
    aLayoutInfo : TcVnArray33_UDINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Source image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLayoutInfo	TcVnArray33_UDINT [▶ 173]	Contains layout information of ipImage. First entry of the aLayoutInfo is the number of dimensions of ipImage (max. 32), followed by the sizes of each dimension.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.44 F_VN_PadImageBorder

Add padding (zeros) around the original image borders.

Syntax

Definition:

```
FUNCTION F_VN_PadImageBorder : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipPaddedImage : Reference To ITcVnImage;
    nTopBorder    : UDINT;
    nBottomBorder : UDINT;
    nLeftBorder   : UDINT;
    nRightBorder  : UDINT;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipPaddedImage	Reference To ITcVnImage [▶ 426]	Padded destination image (Same type as ipSrcImage, an appropriate destination image will be created if required.)
nTopBorder	UDINT	Padding height in pixels above top border
nBottomBorder	UDINT	Padding height in pixels below bottom border
nLeftBorder	UDINT	Padding width in pixels before left border
nRightBorder	UDINT	Padding width in pixels after right border
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

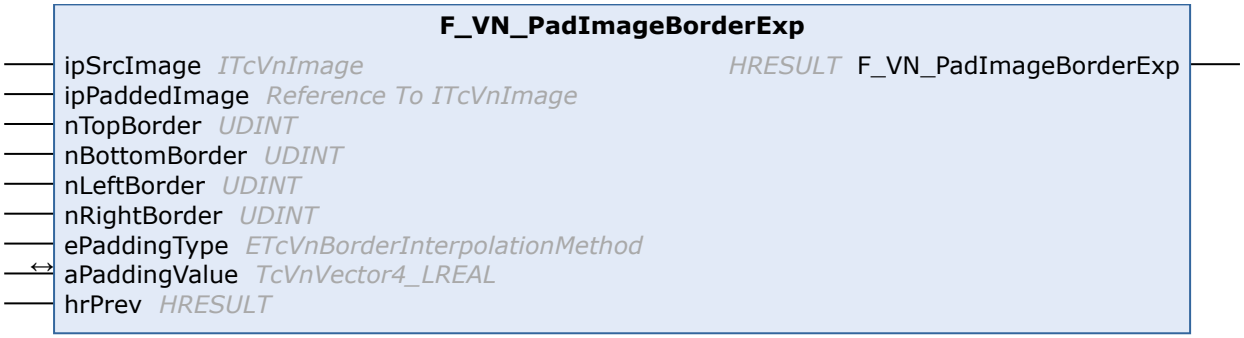
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.45 F_VN_PadImageBorderExp



Add padding around the original image borders. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_PadImageBorderExp : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipPaddedImage : Reference To ITcVnImage;
    nTopBorder    : UDINT;
    nBottomBorder : UDINT;
    nLeftBorder   : UDINT;
    nRightBorder  : UDINT;
    ePaddingType  : ETcVnBorderInterpolationMethod;
END_VAR
VAR_IN_OUT
    aPaddingValue : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipPaddedImage	Reference To ITcVnImage [▶ 426]	Padded destination image (Same type as ipSrcImage, an appropriate destination image will be created if required.)
nTopBorder	UDINT	Padding height in pixels above top border
nBottomBorder	UDINT	Padding height in pixels below bottom border
nLeftBorder	UDINT	Padding width in pixels before left border
nRightBorder	UDINT	Padding width in pixels after right border
ePaddingType	ETcVnBorderInterpolationMethod [▶ 177]	Specifies how the pixel values of the padding area are determined
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aPaddingValue	TcVnVector4_LREAL [▶ 173]	Specifies the padding value if CONSTANT is used

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.5.46 F_VN_ReinterpretUnsupportedImage

Reinterpret an unsupported image using additional information. If successful, the image will be of a supported type.

Syntax

Definition:

```
FUNCTION F_VN_ReinterpretUnsupportedImage : HRESULT
VAR_INPUT
    ipImage    : ITcVnImage;
    ePixelType : ETcVnElementType;
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Image to be reinterpreted
ePixelType	ETcVnElementType [▶ 210]	Pixel element type to be used for reinterpreting the image data
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.6 Code Quality

Diese Gruppe enthält Funktionen zur Qualitätsbewertung von verschiedenen 1D- und 2D-Codes. Die Bewertung erfolgt nach internationalen Standards. Die Prüfung der Codequalität soll sicherstellen, dass der entsprechende Code zum Zeitpunkt der Verwendung zuverlässig dekodiert werden kann.

Um vergleichbare Ergebnisse zu erhalten, definieren die Standards Methoden zur Messung und Bewertung von spezifischen Attributen sowie zur Ableitung einer Gesamtbewertung der Codequalität. Das Ziel ist, die zu erwartende Lesbarkeit der Codes, unter Verwendung von verschiedenen Arten von Lesegeräten, in verschiedenen Anwendungen, Umgebungen und unter unterschiedlichen Bedingungen weitestgehend vorherzusagen. Üblicherweise wird daher für jede Anwendung einzeln eine minimale Qualitätsstufe festgelegt, die akzeptabel ist.

Zum einen beschreiben die Standards Anforderungen, die von den verwendeten Systemen eingehalten werden müssen, um überhaupt verwertbare Ergebnisse zu erhalten. Zum anderen geben die Standards Empfehlungen wie z.B. zur Beleuchtung, zu Objektiveinstellungen und zu dem geometrischen Aufbau, um die Bewertungen mit anderen Systemen vergleichen zu können. Wenn man das Ziel verfolgt, ein System aufzubauen, das dem Standard entspricht, empfiehlt es sich, die entsprechenden Spezifikationen zur Qualitätsbewertung und den zu prüfenden Codes zu beschaffen.

Die Funktionen können auch mit Systemen verwendet werden, die nicht dem Standard entsprechen. Dies hat nur zur Folge, dass die ermittelte Qualität nicht mit anderen Systemen verglichen werden kann. Generell sind kleinere Unterschiede, in den einzelnen Bewertungskategorien, bei unterschiedlichen Systemen zu erwarten.

Nach dem Standard ISO/IEC 15416:2016 kann die Druckqualität von Strichcodes für die Codetypen Code 39, Code 128, EAN 8, EAN 13, ITF, UPC-A, UPC-E mit den folgenden Funktionen ermittelt werden:

- F_VN_GradeBarcode(Exp) [▶ 845]

Nach dem Standard ISO/IEC 15415:2011 kann die Druckqualität von zweidimensionalen Matrixsymbolen der Typen ECC200 Data Matrix-Code und QR-Code mit den folgenden Funktionen ermittelt werden:

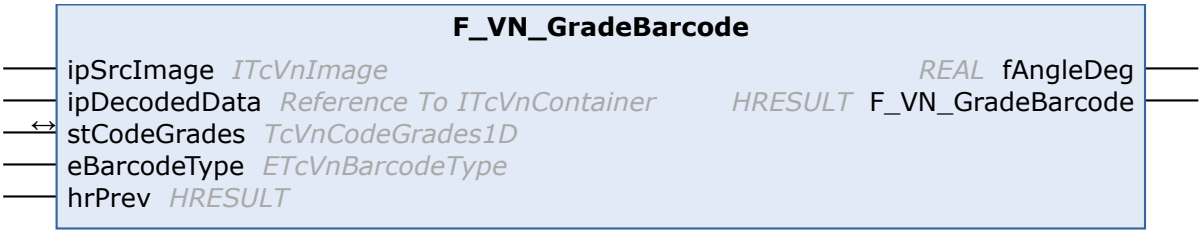
- F_VN_GradeDataMatrixCode(Exp) [▶ 852]
- F_VN_GradeQRCode(Exp) [▶ 857]

Interpretation des HRESULT

Für die Code Grading Funktionen werden die HRESULT [▶ 150]–Rückgabewerte wie folgt benutzt:

Code	Name	Beschreibung
16#000	S_OK	Funktion wurde erfolgreich ausgeführt und alle Bewertungen konnten entsprechend dem Standard durchgeführt werden.
16#001	S_FALSE	Funktion wurde erfolgreich ausgeführt, bei 1D Codes konnten nicht alle Scanlinien dekodiert werden und bei 2D-Codes konnten nicht alle Prüfungen dem Standard entsprechend durchgeführt werden. Die Gesamtnote wurde daher auf 0 gesetzt, die einzelnen Ergebnisse können aber ausgewertet werden.
16#256	S_WATCHDOGTIMEOUT	Funktion wurde vom Watchdog abgebrochen. Je nach Zeitpunkt des Abbruchs können einzelne Ergebnisse ausgewertet werden
16#70C	NOTFOUND	Funktion wurde nicht erfolgreich ausgeführt, bei 1D Codes konnte keine Scanlinie dekodiert werden und bei 2D Codes wurde der Code nicht gefunden bzw. konnte nicht dekodiert werden.
16#7xx	Sämtliche Error-Codes	Funktion wurde nicht erfolgreich ausgeführt. Siehe dazu: ADS Return Codes [▶ 2966]

6.1.4.6.1 F_VN_GradeBarcode



Grades a 1D barcode according to ISO / IEC 15416:2016. Requires uniform illumination, good focus and the image sensor plane parallel to the code plane to achieve meaningful results.

Syntax

Definition:

```
FUNCTION F_VN_GradeBarcode : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipDecodedData : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    stCodeGrades  : TcVnCodeGrades1D;
END_VAR
VAR_INPUT
    eBarcodeType  : ETcVnBarcodeType;
    hrPrev        : HRESULT;
END_VAR
VAR_OUTPUT
    fAngleDeg     : REAL;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image, only containing the horizontally aligned barcode region (USINT, 1 channel, including quiet zones on left and right, excluding Text or empty space on more than 10% total height on top or bottom)
ipDecodedData	Reference To ITcVnContainer [▶ 385]	Returns the decoded data (ContainerType_String_SINT)
eBarcodeType	ETcVnBarcodeType [▶ 175]	Type of the barcode (supported: CODE39, CODE128, EAN8, EAN13, ITF, UPCA, UPCE)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stCodeGrades	TcVnCodeGrades1D [▶ 242]	Returns the code grades in the range from 0 (very bad) to 4 (very good)

Outputs

Name	Type	Description
fAngleDeg	REAL	Returns the approximate clockwise rotation angle in degree

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_GradeBarcode` ermittelt die Druckqualität von Strichcodes nach dem Standard ISO/IEC 15416:2016.

Algorithmus

Die Funktion sucht zunächst nach horizontal ausgerichteten Codes und anschließend nach vertikal ausgerichteten Codes im Bild.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Bild mit Element-Typ `USINT` (8 Bit) sein. Das Eingangsbild darf nur den zu prüfenden Code, einschließlich der entsprechenden Ruhezonen, beinhalten. Der Strichcode muss horizontal oder vertikal ausgerichtet sein und darf keinen Text oder Leerraum auf mehr als 10% der Gesamthöhe ober- oder unterhalb aufweisen. Die Ruhezonen befinden sich bei horizontaler Ausrichtung links und rechts vom Code.

Um aussagekräftige Ergebnisse zu erzielen sind eine gleichmäßige Beleuchtung, nicht unter- oder überbelichtet, eine gute Fokussierung auf die Codeebene und eine dazu parallel ausgerichtete Bildsensorebene erforderlich. Weiterhin muss die Bildauflösung entsprechend der Strichstärke hoch genug sein. Bei Strichcodes sollte die minimale Strichbreite mit mindestens 3 Pixeln abgedeckt werden, um gute Ergebnisse zu erhalten.

Gelesene Daten

Die aus dem Code gelesenen Daten werden als String in dem Container `ipDecodedData` zurückgegeben. Der Container ist dabei vom Typ `ContainerType_String_SINT`. Wurde der Code erfolgreich gefunden und decodiert kann der Inhalt über die Funktion `F_VN_ExportContainer_String` in einen String exportiert werden.

```
hr := F_VN_ExportSubContainer_String(ipDecodedData, sText, nMaxLength, hr);
```

Unterschiedliche Dekodierergebnisse

Die Dekodierung mit dem jeweiligen Referenzalgorithmus kann zu abweichenden Ergebnissen führen. Daher ist es möglich, dass der Codeinhalt von anderen Geräten oder Funktionen gelesen werden kann, der Referenzalgorithmus aber fehlschlägt.

Rückgabe der Qualitätsbewertung

Der Rückgabewert `stCodeGrades` vom Type TcVnCodeGrades1D enthält die ermittelten Einzelergebnisse und ein Gesamtsamtergebnis `fOverall`. Die Qualitätseinstufung erfolgt von 0 (sehr schlecht) bis 4 (sehr gut). Die Ergebnisse werden über mehrere Scanlinien gemittelt.

Parameter	Beschreibung
fDecode	Bewertet, ob der Code mit dem Referenzalgorithmus dekodiert werden konnte.
fSymbolContrast	Bewertet den globalen Symbolkontrast, d.h. die Differenz zwischen dem hellsten und dunkelsten Wert auf der gesamten Scanlinie.
fMinReflectance	Bewertet das Verhältnis vom dunkelsten zum hellsten Wert der Scanlinien.
fMinEdgeContrast	Bewertet den minimalen lokalen Kontrast benachbarter Elemente der Scanlinien.
fModulation	Bewertet das Verhältnis zwischen dem minimalen lokalen Kontrast und dem globalen Symbolkontrast.
fDefects	Bewertet Unregelmäßigkeiten innerhalb von Elementen und Ruhezonen.
fDecodability	Bewertet, wie verlässlich der Code mit dem Referenzalgorithmus dekodiert werden kann.
fOverall	Die Gesamtnote des Codes ist die niedrigste erreichte Note der Einzelprüfungen, gemittelt über alle Scanlinien. Führen verschiedene Scanlinien zu unterschiedlichen dekodierten Daten, ist die Gesamtnote 0.

Barcode-Typ

Der Parameter `eBarcodeType` definiert den Barcode-Typ, dessen Druckqualität beurteilt werden soll. Die folgenden Barcode-Typen aus dem Enum ETcVnBarcodeType [▶ 175] werden unterstützt:

- `TCVN_BT_CODE39`
- `TCVN_BT_CODE128`
- `TCVN_BT_EAN8`
- `TCVN_BT_EAN13`
- `TCVN_BT_ITF`
- `TCVN_BT_UPCA`
- `TCVN_BT_UPCE`

Drehwinkel (Rückgabewert)

`fAngleDeg` gibt den Drehwinkel im Uhrzeigersinn in 90°-Schritten zurück.

Expert-Parameter

Die Expert-Variante F_VN_GradeBarcodeExp [▶ 848] enthält zusätzliche Parameter.

Anwendung

Die Druckqualität von einem Barcode kann z.B. so ermittelt werden:

```
hr := F_VN_GradeBarcode(
    ipSrcImage       := ipSrcImage,
    ipDecodedData    := ipDecodedData,
    stCodeGrades     := stCodeGrades,
    eBarcodeType     := eBarcodeType,
    hrPrev           := hr,
    fAngleDeg        => fAngleDeg);
```

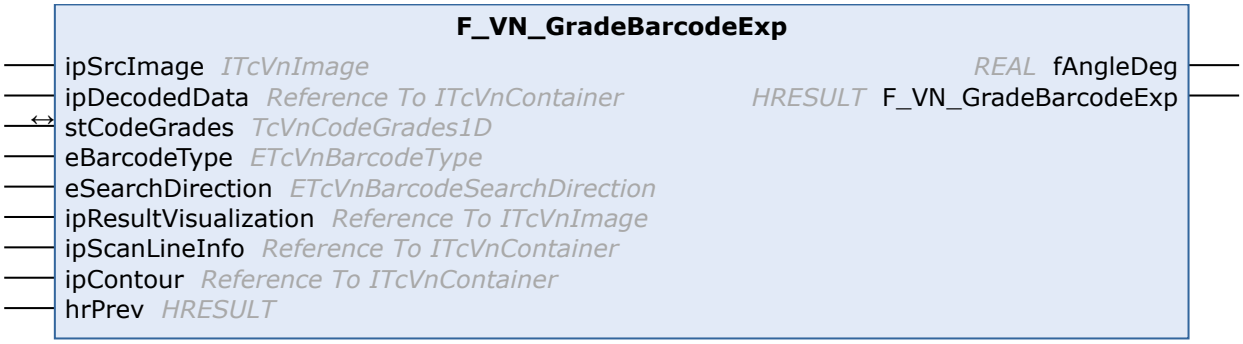
Required License

TC3 Vision Code Quality

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.6.2 F_VN_GradeBarcodeExp



Grades a 1D barcode according to ISO / IEC 15416:2016. Requires uniform illumination, good focus and the image sensor plane parallel to the code plane to achieve meaningful results. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_GradeBarcodeExp : HRESULT
VAR_INPUT
    ipSrcImage            : ITcVnImage;
    ipDecodedData         : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    stCodeGrades          : TcVnCodeGrades1D;
END_VAR
VAR_INPUT
    eBarcodeType          : ETcVnBarcodeType;
    eSearchDirection      : ETcVnBarcodeSearchDirection;
    ipResultVisualization : Reference To ITcVnImage;
    ipScanLineInfo        : Reference To ITcVnContainer;
    ipContour             : Reference To ITcVnContainer;
    hrPrev                : HRESULT;
END_VAR
VAR_OUTPUT
    fAngleDeg             : REAL;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image, only containing the horizontally aligned barcode region (USINT, 1 channel, including quiet zones on left and right, excluding Text or empty space on more than 10% total height on top or bottom)
ipDecodedData	Reference To ITcVnContainer [▶ 385]	Returns the decoded data (ContainerType_String_SINT)
eBarcodeType	ETcVnBarcodeType [▶ 175]	Type of the barcode (supported: CODE39, CODE128, EAN8, EAN13, ITF, UPCA, UPCE)
eSearchDirection	ETcVnBarcodeSearchDirection [▶ 175]	Barcode search direction (BSD_ANY first tries horizontal, then vertical)
ipResultVisualization	Reference To ITcVnImage [▶ 426]	Returns a visualization of the scan lines (optional, set to 0 if not required). Grades are color coded: >= 3 as green, >= 2 as light blue, >= 1 as orange, < 1 as red
ipScanLineInfo	Reference To ITcVnContainer [▶ 385]	Returns additional scan line info for further evaluation, e.g. to use PlotIntensityProfile for visualization (optional, set to 0 if not required; ContainerType_Vector_TcVnVector3_REAL, containing [row index, overall grade, threshold])
ipContour	Reference To ITcVnContainer [▶ 385]	Returns the 4 approximate code corner points, including the quiet zone (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_DINT)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stCodeGrades	TcVnCodeGrades1D [▶ 242]	Returns the code grades in the range from 0 (very bad) to 4 (very good)

Outputs

Name	Type	Description
fAngleDeg	REAL	Returns the approximate clockwise rotation angle in degree

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_GradeBarcodeExp` ist eine Expert-Variante von F_VN_GradeBarcode [▶ 845]. Sie enthält zusätzliche Parameter.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Bild mit Element-Typ `USINT` (8 Bit) sein. Das Eingangsbild darf nur den zu prüfenden Code, einschließlich der entsprechenden Ruhezonen, beinhalten. Der Strichcode muss horizontal oder vertikal ausgerichtet sein und darf keinen Text oder Leerraum auf mehr als 10% der Gesamthöhe ober- oder unterhalb aufweisen. Die Ruhezonen befinden sich bei horizontaler Ausrichtung links und rechts vom Code.

Um aussagekräftige Ergebnisse zu erzielen sind eine gleichmäßige Beleuchtung, nicht unter- oder überbelichtet, eine gute Fokussierung auf die Codeebene und eine dazu parallel ausgerichtete Bildsensorebene erforderlich. Weiterhin muss die Bildauflösung entsprechend der Strichstärke hoch genug sein. Bei Strichcodes sollte die minimale Strichbreite mit mindestens 3 Pixeln abgedeckt werden, um gute Ergebnisse zu erhalten.

Gelesene Daten

Die aus dem Code gelesenen Daten werden als String in dem Container `ipDecodedData` zurückgegeben. Der Container ist dabei vom Typ `ContainerType_String_SINT`. Wurde der Code erfolgreich gefunden und decodiert kann der Inhalt über die Funktion `F_VN_ExportContainer_String` in einen String exportiert werden.

```
hr := F_VN_ExportSubContainer_String(ipDecodedData, sText, nMaxLength, hr);
```

> **Unterschiedliche Dekodierergebnisse**
>
> Die Dekodierung mit dem jeweiligen Referenzalgorithmus kann zu abweichenden Ergebnissen führen. Daher ist es möglich, dass der Codeinhalt von anderen Geräten oder Funktionen gelesen werden kann, der Referenzalgorithmus aber fehlschlägt.

Rückgabe der Qualitätsbewertung

Der Rückgabewert `stCodeGrades` vom Type TcVnCodeGrades1D enthält die ermittelten Einzelergebnisse und ein Gesamtsamtergebnis `fOverall`. Die Qualitätseinstufung erfolgt von 0 (sehr schlecht) bis 4 (sehr gut). Die Ergebnisse werden über mehrere Scanlinien gemittelt.

Parameter	Beschreibung
fDecode	Bewertet, ob der Code mit dem Referenzalgorithmus dekodiert werden konnte.
fSymbolContrast	Bewertet den globalen Symbolkontrast, d.h. die Differenz zwischen dem hellsten und dunkelsten Wert auf der gesamten Scanlinie.
fMinReflectance	Bewertet das Verhältnis vom dunkelsten zum hellsten Wert der Scanlinien.
fMinEdgeContrast	Bewertet den minimalen lokalen Kontrast benachbarter Elemente der Scanlinien.
fModulation	Bewertet das Verhältnis zwischen dem minimalen lokalen Kontrast und dem globalen Symbolkontrast.
fDefects	Bewertet Unregelmäßigkeiten innerhalb von Elementen und Ruhezonen.
fDecodability	Bewertet, wie verlässlich der Code mit dem Referenzalgorithmus dekodiert werden kann.
fOverall	Die Gesamtnote des Codes ist die niedrigste erreichte Note der Einzelprüfungen, gemittelt über alle Scanlinien. Führen verschiedene Scanlinien zu unterschiedlichen dekodierten Daten, ist die Gesamtnote 0.

Barcode-Typ

Der Parameter `eBarcodeType` definiert den Barcode-Typ, dessen Druckqualität beurteilt werden soll. Die folgenden Barcode-Typen aus dem Enum ETcVnBarcodeType [▶ 175] werden unterstützt:

- `TCVN_BT_CODE39`
- `TCVN_BT_CODE128`
- `TCVN_BT_EAN8`
- `TCVN_BT_EAN13`
- `TCVN_BT_ITF`
- `TCVN_BT_UPCA`
- `TCVN_BT_UPCE`

Suchrichtung

Über `eSearchDirection` kann die Suchrichtung vorgegeben werden:

- Bei `TCVN_BSD_ANY` wird zunächst in horizontaler Richtung und anschließend in vertikaler Richtung gesucht.
- Bei `TCVN_BSD_HORIZONTAL` wird nur in horizontaler Richtung gesucht.
- Bei `TCVN_BSD_VERTICAL` wird nur in vertikaler Richtung gesucht.

Ergebnisdarstellung

`ipResultVisualization` gibt ein Bild mit der Visualisierung der Scanlinien und den jeweiligen Bewertungsergebnissen zurück. Die Linien sind entsprechend der einzelnen Bewertungen farblich kodiert: `>= 3` als grün, `>= 2` als hellblau, `>= 1` als orange, `< 1` als rot. Dieser Parameter ist optional und kann auf 0 gesetzt werden, wenn er nicht benötigt wird.

Scanlinien Informationen

`ipScanLineInfo` liefert zusätzliche Informationen über die einzelnen Scanlinien für eine detailliertere Auswertung zurück. Die Rückgabewerte sind vom Typ `ContainerType_Vector_TcVnVector3_REAL` und geben pro Scanlinie je nach Ausrichtung den Index der Bildreihe oder Bildspalte, die Gesamtnote und den ermittelten globalen Threshold an. Die einzelnen Linien können z. B. mit F_VN_PlotIntensityProfile [▶ 1080] visualisiert werden. Dieser Parameter ist optional und kann auf 0 gesetzt werden, wenn er nicht benötigt wird.

Position des Codes

Der Parameter `ipContour` gibt die Code-Eckpunkte, einschließlich der Ruhezone, als vier Konturpunkte vom Typ `ContainerType_Vector_TcVnPoint2_DINT` zurück. Die Berechnung basiert nur auf den Scanlinien, die dekodiert werden konnten. Die Kontur kann z.B. direkt mit F_VN_DrawContours [▶ 1040] gezeichnet oder mit den Funktionen zur Contour Analysis [▶ 985] weiter analysiert werden. Dieser Parameter ist optional und kann auf 0 gesetzt werden, wenn er nicht benötigt wird.

Drehwinkel (Rückgabewert)

`fAngleDeg` gibt den Drehwinkel im Uhrzeigersinn in 90°-Schritten zurück.

Anwendung

Die Druckqualität von einem Barcode kann z.B. so ermittelt werden:

```
hr := F_VN_GradeBarcodeExp(
    ipSrcImage            := ipSrcImage,
    ipDecodedData         := ipDecodedData,
    stCodeGrades          := stCodeGrades,
    eBarcodeType          := eBarcodeType,
    eSearchDirection      := eSearchDirection,
    ipResultVisualization := ipResultVisualization,
    ipScanLineInfo        := ipScanLineInfo,
    ipContour             := ipContour,
    hrPrev                := hr,
    fAngleDeg             => fAngleDeg);
```

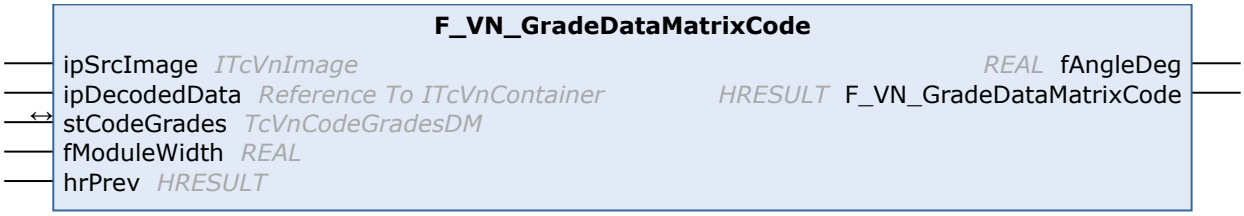
Required License

TC3 Vision Code Quality

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.6.3 F_VN_GradeDataMatrixCode



Grades a Data Matrix code (ECC200) according to ISO / IEC 15415:2011. Requires uniform illumination, good focus and the image sensor plane parallel to the code plane to achieve meaningful results.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_GradeDataMatrixCode : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipDecodedData : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    stCodeGrades  : TcVnCodeGradesDM;
END_VAR
VAR_INPUT
    fModuleWidth  : REAL;
    hrPrev        : HRESULT;
END_VAR
VAR_OUTPUT
    fAngleDeg     : REAL;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image, only containing the Data Matrix code region (USINT, 1 channel, including quiet zone)
ipDecodedData	Reference To ITcVnContainer [▶ 385]	Returns the decoded data (ContainerType_String_SINT)
fModuleWidth	REAL	Minimum module width of the code in the image (in pixels). Must be at least 3, preferably 5 - 8.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stCodeGrades	TcVnCodeGradesDM [▶ 242]	Returns the code grades in the range from 0 (very bad) to 4 (very good)

Outputs

Name	Type	Description
fAngleDeg	REAL	Returns the clockwise rotation angle in degree

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_GradeDataMatrixCode` ermittelt die Druckqualität von ECC200 Data Matrix Codes nach dem Standard ISO/IEC 15415:2011.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Bild mit Element-Typ `USINT` (8 Bit) sein. Das Eingangsbild darf nur den zu prüfenden Code, einschließlich der entsprechenden Ruhezonen, beinhalten.

Um aussagekräftige Ergebnisse zu erzielen sind eine gleichmäßige Beleuchtung, nicht unter- oder überbelichtet, eine gute Fokussierung auf die Codeebene und eine dazu parallel ausgerichtete Bildsensorebene erforderlich. Weiterhin muss die Bildauflösung entsprechend der Modulgröße hoch genug sein. Bei 2D-Codes sollte das kleinste Modul mit mindestens 5 Pixeln abgedeckt werden, um gute Ergebnisse zu erhalten.

Gelesene Daten

Die aus dem Code gelesenen Daten werden als String in dem Container `ipDecodedData` zurückgegeben. Der Container ist dabei vom Typ `ContainerType_String_SINT`. Wurde der Code erfolgreich gefunden und decodiert kann der Inhalt über die Funktion `F_VN_ExportContainer_String` in einen String exportiert werden.

```
hr := F_VN_ExportSubContainer_String(ipDecodedData, sText, nMaxLength, hr);
```

Unterschiedliche Dekodierergebnisse

Die Dekodierung mit dem jeweiligen Referenzalgorithmus kann zu abweichenden Ergebnissen führen. Daher ist es möglich, dass der Codeinhalt von anderen Geräten oder Funktionen gelesen werden kann, der Referenzalgorithmus aber fehlschlägt.

Rückgabe der Qualitätsbewertung

Der Rückgabewert `stCodeGrades` vom Type TcVnCodeGradesDM enthält die ermittelten Einzelergebnisse und ein Gesamtsamtergebnis `nOverall`. Die Qualitätseinstufung erfolgt von 0 (sehr schlecht) bis 4 (sehr gut).

Parameter	Beschreibung
nDecode	Bewertet, ob der Code mit dem Referenzalgorithmus dekodiert werden konnte.
nSymbolContrast	Bewertet den globalen Symbolkontrast, d.h. die Differenz zwischen dem hellsten und dunkelsten Wert im Prüfbereich.
nModulation	Bewertet die Gleichmäßigkeit der dunklen bzw. hellen Module.
nReflectanceMargin	Bewertet, wie gut jedes Modul im Vergleich zum globalen Schwellenwert korrekt als hell oder dunkel unterschieden werden kann.
nFixedPatternDamage	Bewertet die verschiedenen fixen Codebestandteile wie z.B. das Finder- oder Alignment Pattern.
nAxialNonuniformity	Bewertet die gleichmäßige Skalierung des Codes.
nGridNonuniformity	Bewertet Abweichungen des Codegitters.
nUnusedErrorCorrection	Bewertet die verbleibende Kapazität der Fehlerkorrektur.
nOverall	Gesamtnote des Codes, ist die niedrigste erreichte Note der Einzelprüfungen.

Mindestmodulbreite

Mit `fModuleWidth` wird die erwartete Mindestmodulbreite des Codes in Pixel vorgegeben. Wenn die Modulgröße im Bild stärker davon abweicht, kann dies zu ungenauen oder gar keinen Ergebnissen und einer erhöhten Rechenzeit führen.

Drehwinkel (Rückgabewert)

`fAngleDeg` gibt den Drehwinkel im Uhrzeigersinn in Grad zurück.

Expert-Parameter

Die Expert-Variante F_VN_GradeDataMatrixCodeExp [▶ 854] enthält zusätzliche Parameter.

Anwendung

Die Druckqualität von einem Data Matrix Code kann z.B. so ermittelt werden:

```
hr := F_VN_GradeDataMatrixCode(
    ipSrcImage       := ipSrcImage,
    ipDecodedData    := ipDecodedData,
    stCodeGrades     := stCodeGrades,
    fModuleWidth     := fModuleWidth,
    hrPrev           := hr,
    fAngleDeg        => fAngleDeg);
```

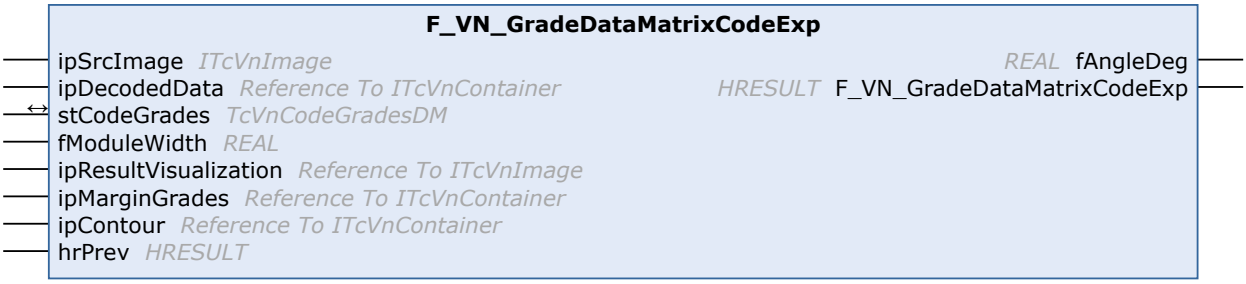
Required License

TC3 Vision Code Quality

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.6.4 F_VN_GradeDataMatrixCodeExp



Grades a Data Matrix code (ECC200) according to ISO / IEC 15415:2011. Requires uniform illumination, good focus and the image sensor plane parallel to the code plane to achieve meaningful results. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_GradeDataMatrixCodeExp : HRESULT
VAR_INPUT
    ipSrcImage            : ITcVnImage;
    ipDecodedData         : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    stCodeGrades          : TcVnCodeGradesDM;
END_VAR
VAR_INPUT
    fModuleWidth          : REAL;
    ipResultVisualization : Reference To ITcVnImage;
    ipMarginGrades        : Reference To ITcVnContainer;
    ipContour             : Reference To ITcVnContainer;
    hrPrev                : HRESULT;
END_VAR
```

```
VAR_OUTPUT
    fAngleDeg             : REAL;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image, only containing the Data Matrix code region (USINT, 1 channel, including quiet zone)
ipDecodedData	Reference To ITcVnContainer [▶ 385]	Returns the decoded data (ContainerType_String_SINT)
fModuleWidth	REAL	Minimum module width of the code in the image (in pixels). Must be at least 3, preferably 5 - 8.
ipResultVisualization	Reference To ITcVnImage [▶ 426]	Returns a visualization of the scan grid (optional, set to 0 if not required). Module margin grades are color coded: >= 3 as green, >= 2 as light blue, >= 1 as orange, < 1 as red
ipMarginGrades	Reference To ITcVnContainer [▶ 385]	Returns the individual margin grade for each module, e.g. to create a custom visualization (optional, set to 0 if not required; ContainerType_Vector_TcVnVector3_REAL, containing [x-position, y-position, grade])
ipContour	Reference To ITcVnContainer [▶ 385]	Returns the 4 code corner points, excluding the quiet zone (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_DINT)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stCodeGrades	TcVnCodeGradesDM [▶ 242]	Returns the code grades in the range from 0 (very bad) to 4 (very good)

Outputs

Name	Type	Description
fAngleDeg	REAL	Returns the clockwise rotation angle in degree

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_GradeDataMatrixCodeExp` ist eine Expert-Variante von F_VN_GradeDataMatrixCode [▶ 852]. Sie enthält zusätzliche Parameter.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Bild mit Element-Typ `USINT` (8 Bit) sein. Das Eingangsbild darf nur den zu prüfenden Code, einschließlich der entsprechenden Ruhezonen, beinhalten.

Um aussagekräftige Ergebnisse zu erzielen sind eine gleichmäßige Beleuchtung, nicht unter- oder überbelichtet, eine gute Fokussierung auf die Codeebene und eine dazu parallel ausgerichtete Bildsensorebene erforderlich. Weiterhin muss die Bildauflösung entsprechend der Modulgröße hoch genug sein. Bei 2D-Codes sollte das kleinste Modul mit mindestens 5 Pixeln abgedeckt werden, um gute Ergebnisse zu erhalten.

Gelesene Daten

Die aus dem Code gelesenen Daten werden als String in dem Container `ipDecodedData` zurückgegeben. Der Container ist dabei vom Typ `ContainerType_String_SINT`. Wurde der Code erfolgreich gefunden und decodiert kann der Inhalt über die Funktion `F_VN_ExportContainer_String` in einen String exportiert werden.

```
hr := F_VN_ExportSubContainer_String(ipDecodedData, sText, nMaxLength, hr);
```

Unterschiedliche Dekodierergebnisse

Die Dekodierung mit dem jeweiligen Referenzalgorithmus kann zu abweichenden Ergebnissen führen. Daher ist es möglich, dass der Codeinhalt von anderen Geräten oder Funktionen gelesen werden kann, der Referenzalgorithmus aber fehlschlägt.

Rückgabe der Qualitätsbewertung

Der Rückgabewert `stCodeGrades` vom Type TcVnCodeGradesDM enthält die ermittelten Einzelergebnisse und ein Gesamtsamtergebnis `nOverall`. Die Qualitätseinstufung erfolgt von 0 (sehr schlecht) bis 4 (sehr gut).

Parameter	Beschreibung
nDecode	Bewertet, ob der Code mit dem Referenzalgorithmus dekodiert werden konnte.
nSymbolContrast	Bewertet den globalen Symbolkontrast, d.h. die Differenz zwischen dem hellsten und dunkelsten Wert im Prüfbereich.
nModulation	Bewertet die Gleichmäßigkeit der dunklen bzw. hellen Module.
nReflectanceMargin	Bewertet, wie gut jedes Modul im Vergleich zum globalen Schwellenwert korrekt als hell oder dunkel unterschieden werden kann.
nFixedPatternDamage	Bewertet die verschiedenen fixen Codebestandteile wie z.B. das Finder- oder Alignment Pattern.
nAxialNonuniformity	Bewertet die gleichmäßige Skalierung des Codes.
nGridNonuniformity	Bewertet Abweichungen des Codegitters.
nUnusedErrorCorrection	Bewertet die verbleibende Kapazität der Fehlerkorrektur.
nOverall	Gesamtnote des Codes, ist die niedrigste erreichte Note der Einzelprüfungen.

Mindestmodulbreite

Mit `fModuleWidth` wird die erwartete Mindestmodulbreite des Codes in Pixel vorgegeben. Wenn die Modulgröße im Bild stärker davon abweicht, kann dies zu ungenauen oder gar keinen Ergebnissen und einer erhöhten Rechenzeit führen.

Ergebnisdarstellung

`ipResultVisualization` gibt ein Bild mit der Visualisierung des Scanrasters und den jeweiligen Bewertungsergebnissen zurück. Die Module sind entsprechend der einzelnen Bewertungen farblich kodiert: `>= 3` als grün, `>= 2` als hellblau, `>= 1` als orange, `< 1` als rot. Dieser Parameter ist optional und kann auf 0 gesetzt werden, wenn er nicht benötigt wird.

Margin Informationen

`ipMarginGrades` liefert zusätzliche Informationen zu den individuellen Margin Bewertungen jedes Moduls für eine detailliertere Auswertung zurück. Der Rückgabevektor vom Typ `ContainerType_Vector_TcVnVector3_REAL` beinhaltet die X- und Y-Position der Mittelpunkte und die Note für jedes Modul. Dieser Parameter ist optional und kann auf 0 gesetzt werden, wenn er nicht benötigt wird.

Position des Codes

Der Parameter `ipContour` gibt die Code-Eckpunkte als vier Konturpunkte vom Typ `ContainerType_Vector_TcVnPoint2_DINT` zurück. Die Eckpunkte liegen am Rand der äußeren Module. Die Kontur kann z.B. direkt mit F_VN_DrawContours [▶ 1040] gezeichnet oder mit den Funktionen zur Contour Analysis [▶ 985] weiter analysiert werden. Dieser Parameter ist optional und kann auf 0 gesetzt werden, wenn er nicht benötigt wird.

Drehwinkel (Rückgabewert)

`fAngleDeg` gibt den Drehwinkel im Uhrzeigersinn in Grad zurück.

Anwendung

Die Druckqualität von einem Data Matrix Code kann z.B. so ermittelt werden:

```
hr := F_VN_GradeDataMatrixCodeExp(
    ipSrcImage              := ipSrcImage,
    ipDecodedData           := ipDecodedData,
    stCodeGrades            := stCodeGrades,
    fModuleWidth            := fModuleWidth,
    ipResultVisualization   := ipResultVisualization,
    ipMarginGrades          := ipMarginGrades,
    ipContour               := ipContour,
    hrPrev                  := hr,
    fAngleDeg               => fAngleDeg);
```

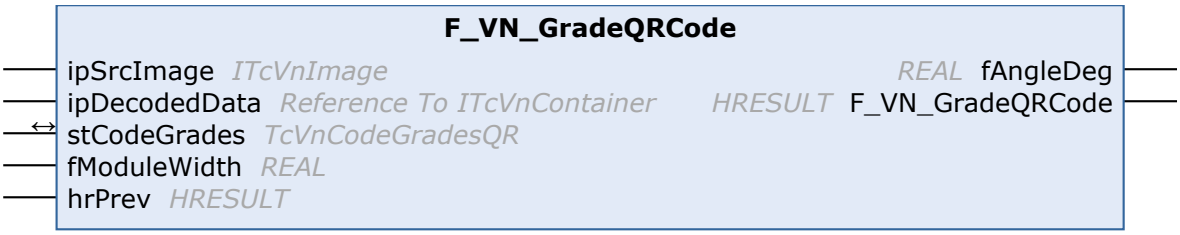
Required License

TC3 Vision Code Quality

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.6.5 F_VN_GradeQRCode



Grades a QR code according to ISO / IEC 15415:2011. Requires uniform illumination, good focus and the image sensor plane parallel to the code plane to achieve meaningful results.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_GradeQRCode : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipDecodedData : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    stCodeGrades  : TcVnCodeGradesQR;
END_VAR
VAR_INPUT
    fModuleWidth  : REAL;
    hrPrev        : HRESULT;
END_VAR
VAR_OUTPUT
    fAngleDeg     : REAL;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image, only containing the QR code region (USINT, 1 channel, including quiet zone)
ipDecodedData	Reference To ITcVnContainer [▶ 385]	Returns the decoded data (ContainerType_String_SINT)
fModuleWidth	REAL	Minimum module width of the code in the image (in pixels). Must be at least 3, preferably 5 - 8.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stCodeGrades	TcVnCodeGradesQR [▶ 243]	Returns the code grades in the range from 0 (very bad) to 4 (very good)

Outputs

Name	Type	Description
fAngleDeg	REAL	Returns the clockwise rotation angle in degree

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_GradeQRCode` ermittelt die Druckqualität von QR-Codes nach dem Standard ISO/IEC 15415:2011.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Bild mit Element-Typ `USINT` (8 Bit) sein. Das Eingangsbild darf nur den zu prüfenden Code, einschließlich der entsprechenden Ruhezonen, beinhalten.

Um aussagekräftige Ergebnisse zu erzielen sind eine gleichmäßige Beleuchtung, nicht unter- oder überbelichtet, eine gute Fokussierung auf die Codeebene und eine dazu parallel ausgerichtete Bildsensorebene erforderlich. Weiterhin muss die Bildauflösung entsprechend der Modulgröße hoch genug sein. Bei 2D-Codes sollte das kleinste Modul mit mindestens 5 Pixeln abgedeckt werden, um gute Ergebnisse zu erhalten.

Gelesene Daten

Die aus dem Code gelesenen Daten werden als String in dem Container `ipDecodedData` zurückgegeben. Der Container ist dabei vom Typ `ContainerType_String_SINT`. Wurde der Code erfolgreich gefunden und decodiert kann der Inhalt über die Funktion `F_VN_ExportContainer_String` in einen String exportiert werden.

```
hr := F_VN_ExportSubContainer_String(ipDecodedData, sText, nMaxLength, hr);
```

> **Unterschiedliche Dekodierergebnisse**
>
> Die Dekodierung mit dem jeweiligen Referenzalgorithmus kann zu abweichenden Ergebnissen führen. Daher ist es möglich, dass der Codeinhalt von anderen Geräten oder Funktionen gelesen werden kann, der Referenzalgorithmus aber fehlschlägt.

Rückgabe der Qualitätsbewertung

Der Rückgabewert `stCodeGrades` vom Type TcVnCodeGradesQR enthält die ermittelten Einzelergebnisse und ein Gesamtsamtergebnis `nOverall`. Die Qualitätseinstufung erfolgt von 0 (sehr schlecht) bis 4 (sehr gut).

Parameter	Beschreibung
nDecode	Bewertet, ob der Code mit dem Referenzalgorithmus dekodiert werden konnte.
nSymbolContrast	Bewertet den globalen Symbolkontrast, d.h. die Differenz zwischen dem hellsten und dunkelsten Wert im Prüfbereich.
nModulation	Bewertet die Gleichmäßigkeit der dunklen bzw. hellen Module.
nReflectanceMargin	Bewertet, wie gut jedes Modul im Vergleich zum globalen Schwellenwert korrekt als hell oder dunkel unterschieden werden kann.
nFixedPatternDamage	Bewertet die verschiedenen fixen Codebestandteile wie z.B. das Finder- oder Alignment Pattern.
nAxialNonuniformity	Bewertet die gleichmäßige Skalierung des Codes.
nGridNonuniformity	Bewertet Abweichungen des Codegitters.
nUnusedErrorCorrection	Bewertet die verbleibende Kapazität der Fehlerkorrektur.
nFormatInfo	Bewertet die Formatinformationen, die rund um das Finderpattern codiert sind.
nVersionInfo	Bewertet die Versionsinformationen, die ab der Version 7 im Code enthalten sind.
nOverall	Gesamtnote des Codes, ist die niedrigste erreichte Note der Einzelprüfungen.

Mindestmodulbreite

Mit `fModuleWidth` wird die erwartete Mindestmodulbreite des Codes in Pixel vorgegeben. Wenn die Modulgröße im Bild stärker davon abweicht, kann dies zu ungenauen oder gar keinen Ergebnissen und einer erhöhten Rechenzeit führen.

Drehwinkel (Rückgabewert)

`fAngleDeg` gibt den Drehwinkel im Uhrzeigersinn in Grad zurück.

Expert-Parameter

Die Expert-Variante F_VN_GradeQRCodeExp [▶ 860] enthält zusätzliche Parameter.

Anwendung

Die Druckqualität von einem QR-Code kann z.B. so ermittelt werden:

```
hr := F_VN_GradeQRCode(
    ipSrcImage       := ipSrcImage,
    ipDecodedData    := ipDecodedData,
    stCodeGrades     := stCodeGrades,
```

```
    fModuleWidth     := fModuleWidth,
    hrPrev           := hr,
    fAngleDeg        => fAngleDeg);
```

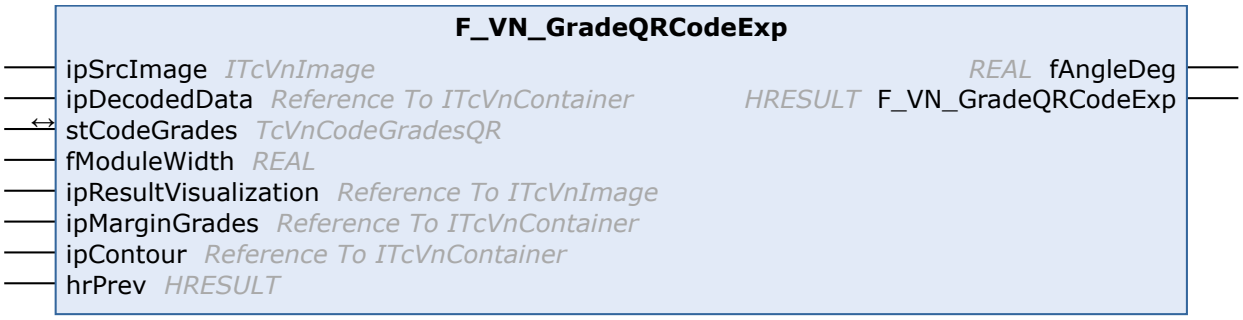
Required License

TC3 Vision Code Quality

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.6.6 F_VN_GradeQRCodeExp



Grades a QR code according to ISO / IEC 15415:2011. Requires uniform illumination, good focus and the image sensor plane parallel to the code plane to achieve meaningful results. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_GradeQRCodeExp : HRESULT
VAR_INPUT
    ipSrcImage            : ITcVnImage;
    ipDecodedData         : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    stCodeGrades          : TcVnCodeGradesQR;
END_VAR
VAR_INPUT
    fModuleWidth          : REAL;
    ipResultVisualization : Reference To ITcVnImage;
    ipMarginGrades        : Reference To ITcVnContainer;
    ipContour             : Reference To ITcVnContainer;
    hrPrev                : HRESULT;
END_VAR
VAR_OUTPUT
    fAngleDeg             : REAL;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image, only containing the QR code region (USINT, 1 channel, including quiet zone)
ipDecodedData	Reference To ITcVnContainer [▶ 385]	Returns the decoded data (ContainerType_String_SINT)
fModuleWidth	REAL	Minimum module width of the code in the image (in pixels). Must be at least 3, preferably 5 - 8.
ipResultVisualization	Reference To ITcVnImage [▶ 426]	Returns a visualization of the scan grid (optional, set to 0 if not required). Module margin grades are color coded: >= 3 as green, >= 2 as light blue, >= 1 as orange, < 1 as red
ipMarginGrades	Reference To ITcVnContainer [▶ 385]	Returns the individual margin grade for each module, e.g. to create a custom visualization (optional, set to 0 if not required; ContainerType_Vector_TcVnVector3_REAL, containing [x-position, y-position, grade])
ipContour	Reference To ITcVnContainer [▶ 385]	Returns the 4 code corner points, excluding the quiet zone (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_DINT)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stCodeGrades	TcVnCodeGradesQR [▶ 243]	Returns the code grades in the range from 0 (very bad) to 4 (very good)

Outputs

Name	Type	Description
fAngleDeg	REAL	Returns the clockwise rotation angle in degree

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_GradeQRCodeExp` ist eine Expert-Variante von F_VN_GradeQRCode [▶ 857]. Sie enthält zusätzliche Parameter.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Bild mit Element-Typ `USINT` (8 Bit) sein. Das Eingangsbild darf nur den zu prüfenden Code, einschließlich der entsprechenden Ruhezonen, beinhalten.

Um aussagekräftige Ergebnisse zu erzielen sind eine gleichmäßige Beleuchtung, nicht unter- oder überbelichtet, eine gute Fokussierung auf die Codeebene und eine dazu parallel ausgerichtete Bildsensorebene erforderlich. Weiterhin muss die Bildauflösung entsprechend der Modulgröße hoch genug sein. Bei 2D-Codes sollte das kleinste Modul mit mindestens 5 Pixeln abgedeckt werden, um gute Ergebnisse zu erhalten.

Gelesene Daten

Die aus dem Code gelesenen Daten werden als String in dem Container `ipDecodedData` zurückgegeben. Der Container ist dabei vom Typ `ContainerType_String_SINT`. Wurde der Code erfolgreich gefunden und decodiert kann der Inhalt über die Funktion `F_VN_ExportContainer_String` in einen String exportiert werden.

```
hr := F_VN_ExportSubContainer_String(ipDecodedData, sText, nMaxLength, hr);
```

> **Unterschiedliche Dekodierergebnisse**
>
> Die Dekodierung mit dem jeweiligen Referenzalgorithmus kann zu abweichenden Ergebnissen führen. Daher ist es möglich, dass der Codeinhalt von anderen Geräten oder Funktionen gelesen werden kann, der Referenzalgorithmus aber fehlschlägt.

Rückgabe der Qualitätsbewertung

Der Rückgabewert `stCodeGrades` vom Type TcVnCodeGradesQR enthält die ermittelten Einzelergebnisse und ein Gesamtsamtergebnis `nOverall`. Die Qualitätseinstufung erfolgt von 0 (sehr schlecht) bis 4 (sehr gut).

Parameter	Beschreibung
nDecode	Bewertet, ob der Code mit dem Referenzalgorithmus dekodiert werden konnte.
nSymbolContrast	Bewertet den globalen Symbolkontrast, d.h. die Differenz zwischen dem hellsten und dunkelsten Wert im Prüfbereich.
nModulation	Bewertet die Gleichmäßigkeit der dunklen bzw. hellen Module.
nReflectanceMargin	Bewertet, wie gut jedes Modul im Vergleich zum globalen Schwellenwert korrekt als hell oder dunkel unterschieden werden kann.
nFixedPatternDamage	Bewertet die verschiedenen fixen Codebestandteile wie z.B. das Finder- oder Alignment Pattern.
nAxialNonuniformity	Bewertet die gleichmäßige Skalierung des Codes.
nGridNonuniformity	Bewertet Abweichungen des Codegitters.
nUnusedErrorCorrection	Bewertet die verbleibende Kapazität der Fehlerkorrektur.
nFormatInfo	Bewertet die Formatinformationen, die rund um das Finderpattern codiert sind.
nVersionInfo	Bewertet die Versionsinformationen, die ab der Version 7 im Code enthalten sind.
nOverall	Gesamtnote des Codes, ist die niedrigste erreichte Note der Einzelprüfungen.

Mindestmodulbreite

Mit `fModuleWidth` wird die erwartete Mindestmodulbreite des Codes in Pixel vorgegeben. Wenn die Modulgröße im Bild stärker davon abweicht, kann dies zu ungenauen oder gar keinen Ergebnissen und einer erhöhten Rechenzeit führen.

Ergebnisdarstellung

`ipResultVisualization` gibt ein Bild mit der Visualisierung des Scanrasters und den jeweiligen Bewertungsergebnissen zurück. Die Module sind entsprechend der einzelnen Bewertungen farblich kodiert: `>= 3` als grün, `>= 2` als hellblau, `>= 1` als orange, `< 1` als rot. Dieser Parameter ist optional und kann auf 0 gesetzt werden, wenn er nicht benötigt wird.

Margin Informationen

`ipMarginGrades` liefert zusätzliche Informationen zu den individuellen Margin Bewertungen jedes Moduls für eine detailliertere Auswertung zurück. Der Rückgabevektor vom Typ `ContainerType_Vector_TcVnVector3_REAL` beinhaltet die X- und Y-Position der Mittelpunkte und die Note für jedes Modul. Dieser Parameter ist optional und kann auf 0 gesetzt werden, wenn er nicht benötigt wird.

Position des Codes

Der Parameter `ipContour` gibt die Code-Eckpunkte als vier Konturpunkte vom Typ `ContainerType_Vector_TcVnPoint2_DINT` zurück. Die Eckpunkte liegen in der Mitte der äußeren Module. Die Kontur kann z.B. direkt mit F_VN_DrawContours [▶ 1040] gezeichnet oder mit den Funktionen zur Contour Analysis [▶ 985] weiter analysiert werden. Dieser Parameter ist optional und kann auf 0 gesetzt werden, wenn er nicht benötigt wird.

Drehwinkel (Rückgabewert)

`fAngleDeg` gibt den Drehwinkel im Uhrzeigersinn in Grad zurück.

Anwendung

Die Druckqualität von einem QR-Code kann z.B. so ermittelt werden:

```
hr := F_VN_GradeQRCodeExp(
    ipSrcImage              := ipSrcImage,
    ipDecodedData           := ipDecodedData,
    stCodeGrades            := stCodeGrades,
    fModuleWidth            := fModuleWidth,
    ipResultVisualization   := ipResultVisualization,
    ipMarginGrades          := ipMarginGrades,
    ipContour               := ipContour,
    hrPrev                  := hr,
    fAngleDeg               => fAngleDeg);
```

Required License

TC3 Vision Code Quality

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.7 Code Reading

Diese Gruppe enthält Funktionen zur Detektion und zum Lesen von 1D- und 2D-Codes.

Funktionen

1D Codes

- F_VN_ReadBarcode(Exp) [▶ 867]
  - CodaBar
  - Code39
  - Code93
  - Code128
  - EAN8
  - EAN13
  - ITF
  - UPCA
  - UPCE
  - Code39Extended
- F_VN_ReadPharmaCode(Exp) [▶ 882]

2D Codes

- F_VN_ReadDataMatrixCode(Exp) [▶ 874]

- F_VN_ReadQRCode(Exp) [▶ 888]
- F_VN_ReadDataMatrixCodeRoi [▶ 877]

Code-Anforderungen

Grundsätzlich wird eine Modulgröße von mindestens drei Pixeln vorausgesetzt; für stabilere Ergebnisse werden mehr als drei Pixel empfohlen. Das Bild sollte scharf sein und einen hohen Kontrast haben. Zudem wird eine Codeausrichtung von ca. 0° oder 90° empfohlen, sodass der Code parallel zu einer Bildachse liegt.

Interpretation des HRESULT

Für die Code Reading Funktionen werden die HRESULT [▶ 150]–Rückgabewerte wie folgt benutzt:

Code	Name	Beschreibung
16#000	S_OK	Funktion wurde erfolgreich ausgeführt und alle erwarteten Codes wurden gefunden (mindestens ein Code bei den Standard-Funktionen und nCodeNumber bei den Experten-Funktionen).
16#001	S_FALSE	Funktion wurde erfolgreich ausgeführt, aber nicht alle erwarteten Codes wurden gefunden (mindestens ein Code bei den Standard-Funktionen und nCodeNumber bei den Experten-Funktionen).
16#256	S_WATCHDOGTIMEOUT	Funktion wurde vom Watchdog abgebrochen. Einige Code Reading Funktionen können Teilergebnisse zurückgeben (siehe dazu einzelne Funktionsbeschreibungen).
16#7xx	Sämtliche Error-Codes	Funktion wurde nicht erfolgreich ausgeführt. Siehe dazu: ADS Return Codes [▶ 2966]

Daher genügt die übliche Überprüfung mit `SUCCEEDED()` nicht, um auf einen gefundenen Code und damit auf vorhandene Daten in `ipDecodedData` zu schließen. Dazu kann folgende Abfrage benutzt werden:

```
IF hr = S_OK THEN
    // Export Code into String
    hr := F_VN_ExportSubContainer_String(ipDecodedCode, 0, sCodeAsString, 255, hr);
    // Use sCodeAsString
ELSIF SUCCEEDED(hr) THEN
    // Process partial results
ELSE
    // Error handling
END_IF
```

Suchstrategien bei 2D-Codes

Für die 2D-Codes werden die Suchalgorithmen auf verschiedene Varianten des Eingangsbildes angewandt, um möglichst viele Codes zu erkennen. Zum Generieren verschiedener Varianten wird Invertieren sowie Spiegeln des Bildes genutzt. Welche Varianten ausprobiert werden, wird als Suchstrategie bezeichnet. Damit Sie mehr Kontrolle über die Laufzeit der Funktionen haben, kann diese Suchstrategie konfiguriert werden. Dafür steht das Enum ETcVn2dCodeSearchStrategy [▶ 173] zur Verfügung. Damit kann für jede mögliche Bildtransformation (Invertierung und Spiegelung) jeweils festgelegt werden, ob sie angewandt werden soll und ob das originale Bild oder das transformierte Bild als erstes auf Codes überprüft werden soll.

Der Funktionsparameter `eSearchStrategy` definiert für entsprechende Code Reading Expertenfunktionen genau diese Suchstrategie. Die folgenden Tabellen zeigen die verschiedenen Kombinationen:

Tab. 1: Invertierung

eSearchStrategy	Originalbild	Invertiertes Bild
TCVN_CSS_ONLY_NOT_INVERTED	1.	-
TCVN_CSS_FIRST_NOT_INVERTED	1.	2.
TCVN_CSS_ONLY_INVERTED	-	1.
TCVN_CSS_FIRST_INVERTED	2.	1.

Tab. 2: Spiegelung

eSearchStrategy	Originalbild	Gespiegeltes Bild
TCVN_CSS_ONLY_NOT_FLIPPED	1.	-
TCVN_CSS_FIRST_NOT_FLIPPED	1.	2.
TCVN_CSS_ONLY_FLIPPED	-	1.
TCVN_CSS_FIRST_FLIPPED	2.	1.

Für jede Transformationsart kann eine Suchstrategie festgelegt werden. Die Suchstrategien der verschiedenen Suchstrategien können wie folgt verknüpft werden:

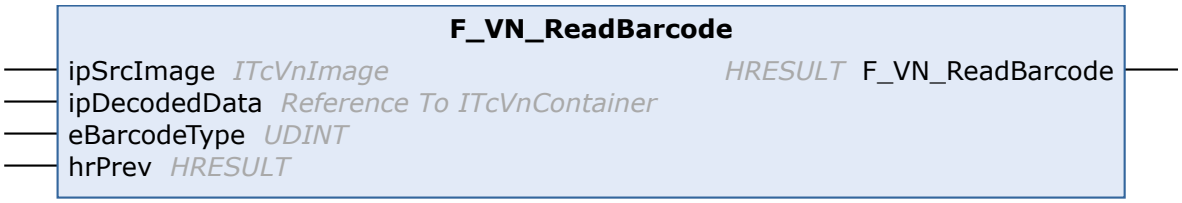
```
eSearchStrategy := TCVN_CSS_ONLY_FLIPPED + TCVN_CSS_FIRST_INVERTED;
```

Zudem kann mit der Option `TCVN_CSS_DEFAULT` eine Standard-Einstellung ausgewählt werden, die je nach Code-Art variiert.

Verknüpfung von Suchstrategien nicht möglich

Eine Verknüpfung mehrerer Suchstrategien derselben Transformationsart ist nicht zulässig. Es kann jeweils nur eine Strategie für Invertierung und Spiegelung miteinander verknüpft werden.

6.1.4.7.1 F_VN_ReadBarcode



Detect and interpret a 1d barcode within the provided image. Can be canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_ReadBarcode : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipDecodedData : Reference To ITcVnContainer;
    eBarcodeType  : UDINT;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT elements, 1 channel or 3 channel (3 channel input is expected to be RGB and internally converted to Gray))
ipDecodedData	Reference To ITcVnContainer [▶ 385]	Returns the decoded code (ContainerType_Vector_String_SINT)
eBarcodeType	UDINT	Types of barcode to search for (ETcVnBarcodeType)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_ReadBarcode` erkennt einen Barcode im Bild und liest diesen.

Algorithmus

Die Funktion sucht zunächst nach horizontal ausgerichteten Codes und anschließend nach vertikal ausgerichteten Codes im Bild.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges oder 3-kanaliges Bild mit Element-Typ `USINT` (8 Bit) sein. 3-kanalige Bilder werden als RGB interpretiert und intern in ein Grauwertbild konvertiert.

Gelesene Daten

Die aus dem Code gelesenen Daten werden als String in dem Container `ipDecodedData` zurückgegeben. Der Container ist dabei vom Typ `ContainerType_Vector_String_SINT`.

Barcode-Typ

Der Parameter `eBarcodeType` definiert Barcode-Typen, nach denen im Bild gesucht werden soll. Alle Barcode-Typen im Enum ETcVnBarcodeType [▶ 175] werden unterstützt. Die Auswahl von mehreren Typen ist möglich. Es wird empfohlen, die Auswahl auf die in der Endanwendung tatsächlich vorkommenden Typen zu beschränken, um die Laufzeit zu verbessern.

```
eBarcodeType := TCVN_BT_EAN13 + TCVN_BT_CODE128;
```

Um nach allen verfügbaren Barcode-Typen zu suchen, verwenden Sie `TCVN_BT_ANY`.

Expert-Parameter

Die Expert-Variante F_VN_ReadBarcodeExp [▶ 867] enthält zusätzliche Parameter.

Daten aus Container lesen

Wurde der Code erfolgreich gefunden und decodiert, wird der Codeinhalt im Container `ipDecodedData` zurückgegeben und kann über die Funktion F_VN_ExportSubContainer_String [▶ 783] in einen String exportiert werden.

```
hr := F_VN_ExportSubContainer_String( ipDecodedData, 0, sText, nMaxLength, hr);
```

Beispiele

- EAN-13 Barcode Reading [▶ 2845]
- Ergebnisauswertung beim Code Reading [▶ 2854]

Verwandte Funktionen

- F_VN_ReadBarcode [▶ 865]
- F_VN_ReadPharmaCode [▶ 880]
- F_VN_ReadDataMatrixCode [▶ 871]
- F_VN_ReadQRCode [▶ 885]

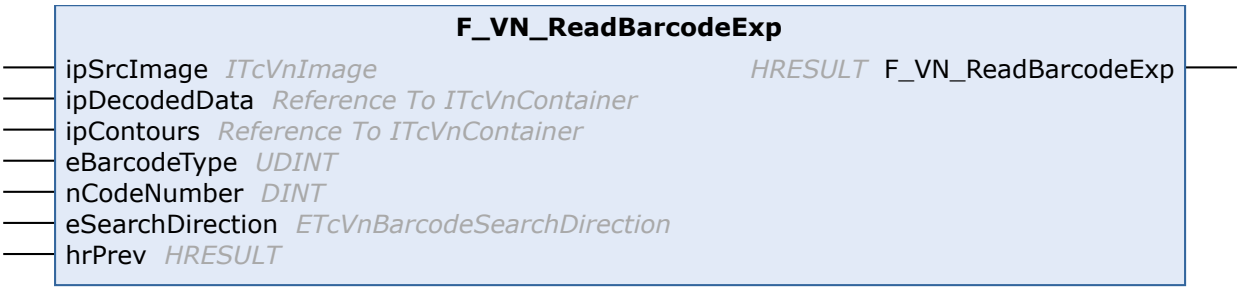
Required License

TC3 Vision Code Reading

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.7.2 F_VN_ReadBarcodeExp



Detect and interpret a 1d barcode within the provided image. Can be canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_ReadBarcodeExp : HRESULT
VAR_INPUT
    ipSrcImage       : ITcVnImage;
    ipDecodedData    : Reference To ITcVnContainer;
    ipContours       : Reference To ITcVnContainer;
    eBarcodeType     : UDINT;
    nCodeNumber      : DINT;
    eSearchDirection : ETcVnBarcodeSearchDirection;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT elements, 1 channel or 3 channel (3 channel input is expected to be RGB and internally converted to Gray))
ipDecodedData	Reference To ITcVnContainer [▶ 385]	Returns the decoded code (ContainerType_Vector_String_SINT)
ipContours	Reference To ITcVnContainer [▶ 385]	Returns the code positions as contours (optional, set to 0 if not required; ContainerType_Vector_Vector_TcVnPoint2_DINT)
eBarcodeType	UDINT	Types of barcode to search for (ETcVnBarcodeType)
nCodeNumber	DINT	Number of 1d barcode that should be detected within the provided image (currently only 1 supported).
eSearchDirection	ETcVnBarcodeSearchDirection [▶ 175]	Barcode search direction.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_ReadBarcodeExp` ist die Expert-Variante von F_VN_ReadBarcode [▶ 865]. Sie enthält zusätzliche Parameter zur Rückgabe der Codeposition und zum Einstellen der Suchrichtung.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges oder 3-kanaliges Bild mit Element-Typ `USINT` (8 Bit) sein. 3-kanalige Bilder werden als RGB interpretiert und intern in ein Grauwertbild konvertiert.

Gelesene Daten

Die aus dem Code gelesenen Daten werden als String in dem Container `ipDecodedData` zurückgegeben. Der Container ist dabei vom Typ `ContainerType_Vector_String_SINT`.

Barcode-Typ

Der Parameter `eBarcodeType` definiert Barcode-Typen, nach denen im Bild gesucht werden soll. Alle Barcode-Typen im Enum ETcVnBarcodeType [▶ 175] werden unterstützt. Die Auswahl von mehreren Typen ist möglich. Es wird empfohlen, die Auswahl auf die in der Endanwendung tatsächlich vorkommenden Typen zu beschränken, um die Laufzeit zu verbessern.

```
eBarcodeType := TCVN_BT_EAN13 + TCVN_BT_CODE128;
```

Um nach allen verfügbaren Barcode-Typen zu suchen, verwenden Sie `TCVN_BT_ANY`.

Position des Codes

Der Parameter `ipContour` gibt die gefundene Codeposition als Kontur zurück.

Anzahl von Codes

Der Parameter `nCodeNumber` definiert, nach wie vielen Codes gesucht werden soll.

Suche nach mehreren Barcodes nicht unterstützt

Die Suche nach mehreren Barcodes in einem Bild wird aktuell nicht unterstützt. Daher ist der Parameter nCodeNumber := 1 zu setzen.

Suchrichtung

Über `eSearchDirection` kann die Suchrichtung vorgegeben werden:

- Bei `TCVN_BSD_ANY` wird zunächst in horizontaler Richtung und anschließend in vertikaler Richtung gesucht.
- Bei `TCVN_BSD_HORIZONTAL` wird nur in horizontaler Richtung gesucht.
- Bei `TCVN_BSD_VERTICAL` wird nur in vertikaler Richtung gesucht.

Beispiele

- EAN-13 Barcode Reading [▶ 2845]
- Ergebnisauswertung beim Code Reading [▶ 2854]

Verwandte Funktionen

- F_VN_ReadBarcode [▶ 865]
- F_VN_ReadPharmaCode [▶ 880]
- F_VN_ReadDataMatrixCode [▶ 871]
- F_VN_ReadQRCode [▶ 885]

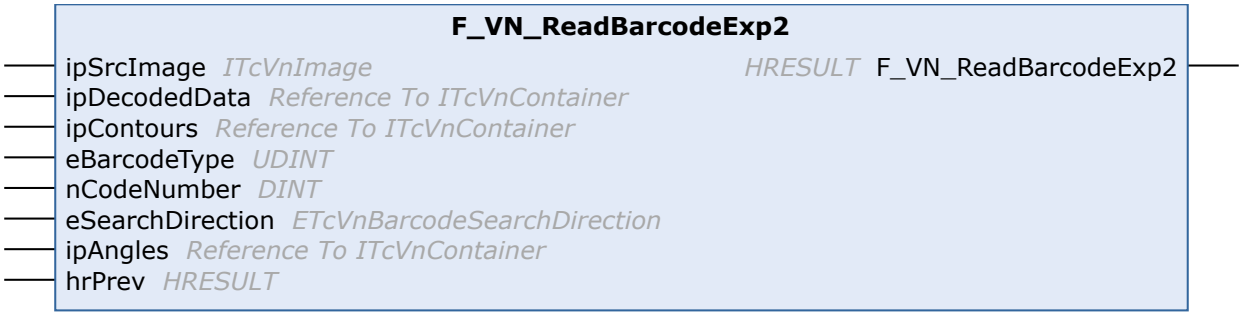
Required License

TC3 Vision Code Reading

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.7.3 F_VN_ReadBarcodeExp2



Detect and interpret a 1d barcode within the provided image. Can be canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_ReadBarcodeExp2 : HRESULT
VAR_INPUT
    ipSrcImage       : ITcVnImage;
    ipDecodedData    : Reference To ITcVnContainer;
    ipContours       : Reference To ITcVnContainer;
    eBarcodeType     : UDINT;
    nCodeNumber      : DINT;
```

```
    eSearchDirection : ETcVnBarcodeSearchDirection;
    ipAngles         : Reference To ITcVnContainer;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT elements, 1 channel or 3 channel (3 channel input is expected to be RGB and internally converted to Gray))
ipDecodedData	Reference To ITcVnContainer [▶ 385]	Returns the decoded code (ContainerType_Vector_String_SINT)
ipContours	Reference To ITcVnContainer [▶ 385]	Returns the code positions as contours (optional, set to 0 if not required; ContainerType_Vector_Vector_TcVnPoint2_DINT)
eBarcodeType	UDINT	Types of barcode to search for (ETcVnBarcodeType)
nCodeNumber	DINT	Number of 1d barcode that should be detected within the provided image (currently only 1 supported).
eSearchDirection	ETcVnBarcodeSearchDirection [▶ 175]	Barcode search direction.
ipAngles	Reference To ITcVnContainer [▶ 385]	Returns the clockwise code rotation angles (optional, set to 0 if not required; ContainerType_Vector_REAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_ReadBarcodeExp` ist die Expert-Variante von F_VN_ReadBarcode [▶ 865]. Sie enthält zusätzliche Parameter zur Rückgabe der Codeposition und zum Einstellen der Suchrichtung.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges oder 3-kanaliges Bild mit Element-Typ `USINT` (8 Bit) sein. 3-kanalige Bilder werden als RGB interpretiert und intern in ein Grauwertbild konvertiert.

Gelesene Daten

Die aus dem Code gelesenen Daten werden als String in dem Container `ipDecodedData` zurückgegeben. Der Container ist dabei vom Typ `ContainerType_Vector_String_SINT`.

Barcode-Typ

Der Parameter `eBarcodeType` definiert Barcode-Typen, nach denen im Bild gesucht werden soll. Alle Barcode-Typen im Enum ETcVnBarcodeType [▶ 175] werden unterstützt. Die Auswahl von mehreren Typen ist möglich. Es wird empfohlen, die Auswahl auf die in der Endanwendung tatsächlich vorkommenden Typen zu beschränken, um die Laufzeit zu verbessern.

```
eBarcodeType := TCVN_BT_EAN13 + TCVN_BT_CODE128;
```

Um nach allen verfügbaren Barcode-Typen zu suchen, verwenden Sie `TCVN_BT_ANY`.

Position des Codes

Der Parameter `ipContour` gibt die gefundene Codeposition als Kontur zurück.

Anzahl von Codes

Der Parameter `nCodeNumber` definiert, nach wie vielen Codes gesucht werden soll.

> **Suche nach mehreren Barcodes nicht unterstützt**
>
> Die Suche nach mehreren Barcodes in einem Bild wird aktuell nicht unterstützt. Daher ist der Parameter nCodeNumber := 1 zu setzen.

Suchrichtung

Über `eSearchDirection` kann die Suchrichtung vorgegeben werden:

- Bei `TCVN_BSD_ANY` wird zunächst in horizontaler Richtung und anschließend in vertikaler Richtung gesucht.
- Bei `TCVN_BSD_HORIZONTAL` wird nur in horizontaler Richtung gesucht.
- Bei `TCVN_BSD_VERTICAL` wird nur in vertikaler Richtung gesucht.

Drehwinkel

Der Parameter `ipAngles` gibt die Drehwinkel der gefundenen Codes im Uhrzeigersinn in 90°-Schritten zurück. Der Container ist vom Typ `ContainerType_Vector_REAL`. Dieser Parameter ist optional und kann auf 0 gesetzt werden, wenn er nicht benötigt wird.

Beispiele

- EAN-13 Barcode Reading [▶ 2845]
- Ergebnisauswertung beim Code Reading [▶ 2854]

Verwandte Funktionen

- F_VN_ReadBarcode [▶ 865]
- F_VN_ReadPharmaCode [▶ 880]
- F_VN_ReadDataMatrixCode [▶ 871]
- F_VN_ReadQRCode [▶ 885]

Required License

TC3 Vision Code Reading

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.7.4 F_VN_ReadDataMatrixCode

Detect and interpret a data matrix code (ECC200) within the provided image.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_ReadDataMatrixCode : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipDecodedData : Reference To ITcVnContainer;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT elements, 1 or 3 channels)
ipDecodedData	Reference To ITcVnContainer [▶ 385]	Returns the decoded code (ContainerType_Vector_String_SINT)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

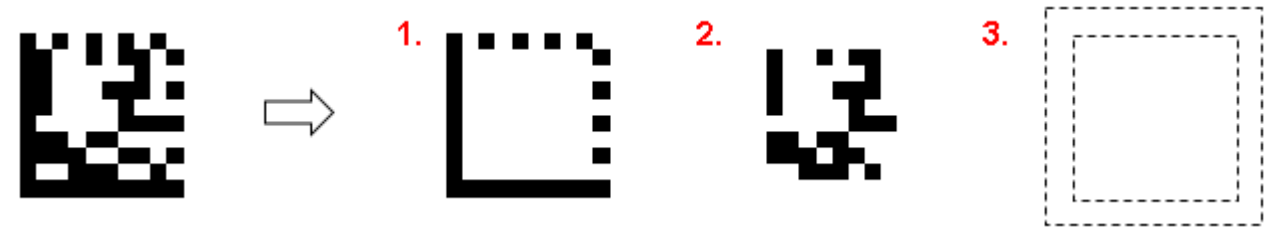
Weiterführende Informationen

Der Data Matrix Barcode nach IEC 16022 ist ein 2D-Code, der es ermöglicht, viele Daten auf kleinstem Raum zu speichern. Dadurch wird dieser oft zur Bauteil- und Produktmarkierung verwendet.

Die Funktion F_VN_ReadDataMatrixCode kann nur ECC 200 Codetypen im Bild erkennen und dekodieren. Dabei wird von einem Code pro Bild ausgegangen. Sollen mehrere Data Matrix Codes in einem Bild erkannt und dekodiert werden ist die Funktion F_VN_ReadDataMatrixCodeExp [▶ 874] zu verwenden.

Data Matrix Code Aufbau

Der Data Matrix Code setzt sich aus Rechtecken bzw. Punkten zusammen, aus denen sich ein quadratisches oder rechteckiges Symbol zusammensetzt, das sich wiederum in drei Bereiche aufteilen lässt:



1. Finder Pattern
   Das durchgezogene L (Solid-Line Locator) dient zur Auffindung und Ausrichtung des Codes, während sich aus den abwechselnd hellen und dunklen Zellen (Pattern-Line Locator) die Zellgröße und Zellanordnung bestimmen lässt.
2. Datenbereich
   Umrandet vom Finder Pattern befindet sich der Datenbereich.
3. Ruhezone
   Die Ruhezone ist ein mind. 1,5 Zellen breiter Bereich um das Finder Pattern herum in der Hintergrundfarbe des Codes.

Störungen / Beschädigungen im Finder Pattern oder in der Ruhezone können eine Detektion des Codes verhindern.

Fehlerkorrektur

Die Data Matrix Code Version ECC 200 (ECC = Error Correction Code) hat sich aufgrund der besseren Fehlerkorrektur durchgesetzt. Neben den eigentlichen Daten sind diese redundant entsprechend des Reed-Solomon-Codes codiert im Code enthalten. Data Matrix Codes dieser Version können von der Funktion dekodiert werden und bei Beschädigung kann eine Fehlerkorrektur durchgeführt werden.

Codesuche

Die Funktion F_VN_ReadDataMatrixCode sucht ausgehend von der mittleren Bildzeile nach einem Data Matrix Code im Bild.

Daten aus Container lesen

Wurde der Code erfolgreich gefunden und decodiert, wird der Codeinhalt im Container `ipDecodedData` zurückgegeben und kann über die Funktion F_VN_ExportSubContainer_String [▶ 783] in einen String exportiert werden.

```
hr := F_VN_ExportSubContainer_String( ipDecodedData, 0, sText, nMaxLength, hr);
```

HRESULT

Wie alle TwinCAT Vision API-Funktionen gibt auch diese Funktion ein HRESULT [▶ 150] zurück, um anzugeben, ob die Ausführung erfolgreich war oder fehlgeschlagen ist. Im Erfolgsfall kann bei der nachfolgenden Ergebnisbehandlung zwischen folgenden Success-Codes unterschieden werden. Um zwischen diesen Fällen zu unterscheiden, kann die `HRESULT`-Variable direkt mit `S_OK` oder `S_FALSE` verglichen werden.

Code	Name	Beschreibung
16#000	S_OK	Die Funktion wurde erfolgreich ausgeführt. Die angegebene Anzahl an Codes wurde im Bild gefunden.
16#001	S_FALSE	Die Funktion wurde erfolgreich ausgeführt. Es wurden weniger Codes als angegeben im Bild gefunden.
16#256	S_WATCHDOG TIMEOUT	Die Funktion wurde vom Watchdog [▶ 156] abgebrochen.

Beispiele

- Data Matrix Code Reading [▶ 2848]
- Ergebnisauswertung beim Code Reading [▶ 2854]

Verwandte Funktionen

- F_VN_ReadDataMatrixCodeExp [▶ 874]
- F_VN_ReadBarcode [▶ 865] bzw. F_VN_ReadBarcodeExp [▶ 867]
- F_VN_ReadPharmaCode [▶ 880] bzw. F_VN_ReadPharmaCodeExp [▶ 882]
- F_VN_ReadQRCode [▶ 885] bzw. F_VN_ReadQRCodeExp [▶ 888]

Required License

TC3 Vision Code Reading

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.7.5 F_VN_ReadDataMatrixCodeExp

Detect and interpret a data matrix code (ECC200) within the provided image.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_ReadDataMatrixCodeExp : HRESULT
VAR_INPUT
    ipSrcImage      : ITcVnImage;
    ipDecodedData   : Reference To ITcVnContainer;
    ipContours      : Reference To ITcVnContainer;
    nCodeNumber     : DINT;
    eSearchStrategy : UDINT;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT elements, 1 or 3 channels)
ipDecodedData	Reference To ITcVnContainer [▶ 385]	Returns the decoded code (ContainerType_Vector_String_SINT)
ipContours	Reference To ITcVnContainer [▶ 385]	Returns the code positions as contours (optional, set to 0 if not required; ContainerType_Vector_Vector_TcVnPoint2_DINT)
nCodeNumber	DINT	Number of data matrix codes that should be detected within the provided image. (If set to -1, it tries to detect all data matrix codes.)
eSearchStrategy	UDINT	Used search strategy (ETcVn2dCodeSearchStrategy)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion F_VN_ReadDataMatrixCodeExp entspricht der Funktion F_VN_ReadDataMatrixCode [▶ 871] erweitert um die folgenden Parameter.

ipContours

Der Parameter `ipContour` gibt die gefundenen Codepositionen jeweils als Kontur zurück.

nCodeNumber

Der Parameter `nCodeNumber` definiert, nach wie vielen Codes gesucht werden soll. Bei *-1* wird so lange gesucht bis kein weiterer Code mehr im Bild gefunden wird oder ein Watchdog die Funktion abbricht.

eSearchStrategy

Der Parameter `eSearchStrategy` definiert, welche Suchstrategien für die Codesuche genutzt werden sollen. Die Standard-Option `TCVN_CSS_DEFAULT` führt zu der Suchstrategie `TCVN_CSS_FIRST_NOT_FLIPPED`. Eine Beschreibung der Suchstrategien finden Sie unter Code Reading Suchstrategien [▶ 864].

Suchstrategien

Zurzeit unterstützt diese Funktion nur die Einstellung der Suchstrategie zur Spiegelung. Eine Invertierung wird immer automatisch vorgenommen, falls im Originalbild nicht alle erwarteten Codes gefunden werden.

Beispiele

- Data Matrix Code Reading [▶ 2848]
- Ergebnisauswertung beim Code Reading [▶ 2854]

Verwandte Funktionen

- F_VN_ReadDataMatrixCode [▶ 871]
- F_VN_ReadBarcode [▶ 865] bzw. F_VN_ReadBarcodeExp [▶ 867]
- F_VN_ReadPharmaCode [▶ 880] bzw. F_VN_ReadPharmaCodeExp [▶ 882]
- F_VN_ReadQRCode [▶ 885] bzw. F_VN_ReadQRCodeExp [▶ 888]

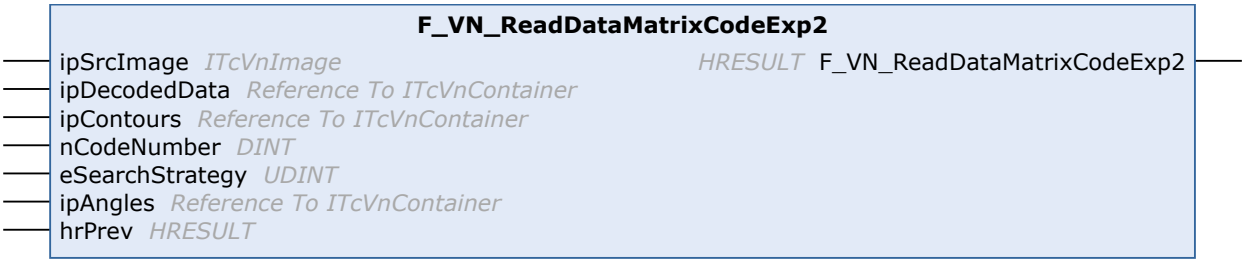
Required License

TC3 Vision Code Reading

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.7.6 F_VN_ReadDataMatrixCodeExp2



Detect and interpret a data matrix code (ECC200) within the provided image.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_ReadDataMatrixCodeExp2 : HRESULT
VAR_INPUT
    ipSrcImage      : ITcVnImage;
    ipDecodedData   : Reference To ITcVnContainer;
    ipContours      : Reference To ITcVnContainer;
    nCodeNumber     : DINT;
    eSearchStrategy : UDINT;
    ipAngles        : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT elements, 1 or 3 channels)
ipDecodedData	Reference To ITcVnContainer [▶ 385]	Returns the decoded code (ContainerType_Vector_String_SINT)
ipContours	Reference To ITcVnContainer [▶ 385]	Returns the code positions as contours (optional, set to 0 if not required; ContainerType_Vector_Vector_TcVnPoint2_DINT)
nCodeNumber	DINT	Number of data matrix codes that should be detected within the provided image. (If set to -1, it tries to detect all data matrix codes.)
eSearchStrategy	UDINT	Used search strategy (ETcVn2dCodeSearchStrategy)
ipAngles	Reference To ITcVnContainer [▶ 385]	Returns the clockwise code rotation angles (optional, set to 0 if not required; ContainerType_Vector_REAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion F_VN_ReadDataMatrixCodeExp entspricht der Funktion F_VN_ReadDataMatrixCode [▶ 871] erweitert um die folgenden Parameter.

ipContours

Der Parameter `ipContour` gibt die gefundenen Codepositionen jeweils als Kontur zurück.

nCodeNumber

Der Parameter `nCodeNumber` definiert, nach wie vielen Codes gesucht werden soll. Bei *-1* wird so lange gesucht bis kein weiterer Code mehr im Bild gefunden wird oder ein Watchdog die Funktion abbricht.

eSearchStrategy

Der Parameter `eSearchStrategy` definiert, welche Suchstrategien für die Codesuche genutzt werden sollen. Die Standard-Option `TCVN_CSS_DEFAULT` führt zu der Suchstrategie `TCVN_CSS_FIRST_NOT_FLIPPED`. Eine Beschreibung der Suchstrategien finden Sie unter Code Reading Suchstrategien [▶ 864].

Suchstrategien

Zurzeit unterstützt diese Funktion nur die Einstellung der Suchstrategie zur Spiegelung. Eine Invertierung wird immer automatisch vorgenommen, falls im Originalbild nicht alle erwarteten Codes gefunden werden.

Drehwinkel

Der Parameter `ipAngles` gibt die Drehwinkel der gefundenen Codes im Uhrzeigersinn in Grad-Schritten zurück. Der Container ist vom Typ `ContainerType_Vector_REAL`. Dieser Parameter ist optional und kann auf 0 gesetzt werden, wenn er nicht benötigt wird.

Beispiele

- Data Matrix Code Reading [▶ 2848]
- Ergebnisauswertung beim Code Reading [▶ 2854]

Verwandte Funktionen

- F_VN_ReadDataMatrixCode [▶ 871]
- F_VN_ReadBarcode [▶ 865] bzw. F_VN_ReadBarcodeExp [▶ 867]
- F_VN_ReadPharmaCode [▶ 880] bzw. F_VN_ReadPharmaCodeExp [▶ 882]
- F_VN_ReadQRCode [▶ 885] bzw. F_VN_ReadQRCodeExp [▶ 888]

Required License

TC3 Vision Code Reading

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.7.7 F_VN_ReadDataMatrixCodeRoi

Detect and interpret a data matrix code (ECC200) within the provided image. The code should be in the image center.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_ReadDataMatrixCodeRoi : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipDecodedData : Reference To ITcVnContainer;
    fModuleWidth  : REAL;
    hrPrev        : HRESULT;
END_VAR
VAR_OUTPUT
    fAngleDeg     : REAL;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT elements, 1 channel)
ipDecodedData	Reference To ITcVnContainer [▶ 385]	Returns the decoded code (ContainerType_String_SINT)
fModuleWidth	REAL	Minimum module width of the code in the image (in pixels). Must be at least 3, preferably 5 - 8.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Outputs

Name	Type	Description
fAngleDeg	REAL	Returns the clockwise rotation angle in degree

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_ReadDataMatrixCodeRoi` ist für die Erkennung eines einzelnen Data Matrix Codes (ECC200) innerhalb eines definierten Bildbereiches optimiert. Dafür nutzt die Funktion einen angepassten Algorithmus, der eine zentrierte Positionierung des Codes im übergebenen Bild voraussetzt.

fModuleWidth

Der Parameter `fModuleWidth` definiert die minimal erwartete Modulbreite des zu lesenden Data Matrix Codes in Pixeln. Ein Modul entspricht einem schwarzen bzw. weißen Quadrat des Codes. Weicht die Modulgröße im Bild stärker davon ab, kann dies zu einer erhöhten Rechenzeit oder keinem Ergebnis führen.

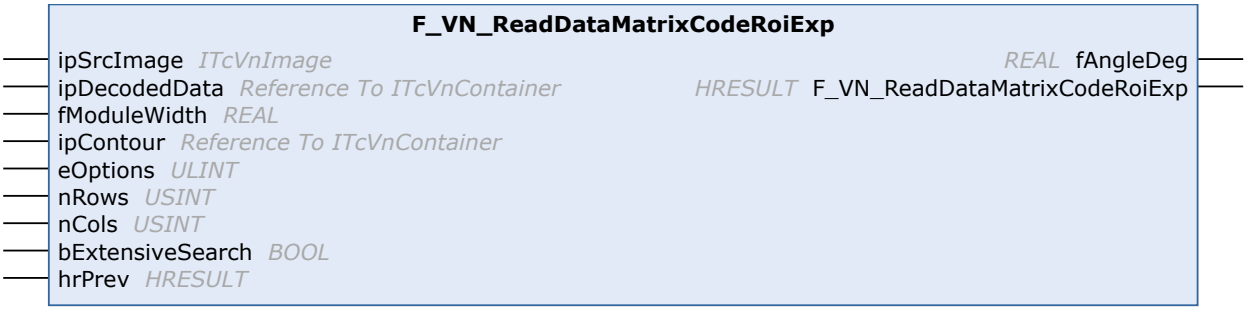
Required License

TC3 Vision Code Reading

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.7.8 F_VN_ReadDataMatrixCodeRoiExp



Detect and interpret a data matrix code (ECC200) within the provided image. The code should be in the image center. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_ReadDataMatrixCodeRoiExp : HRESULT
VAR_INPUT
    ipSrcImage       : ITcVnImage;
    ipDecodedData    : Reference To ITcVnContainer;
    fModuleWidth     : REAL;
    ipContour        : Reference To ITcVnContainer;
    eOptions         : ULINT;
    nRows            : USINT;
    nCols            : USINT;
    bExtensiveSearch : BOOL;
    hrPrev           : HRESULT;
END_VAR
VAR_OUTPUT
    fAngleDeg        : REAL;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT elements, 1 channel)
ipDecodedData	Reference To ITcVnContainer [▶ 385]	Returns the decoded code (ContainerType_String_SINT)
fModuleWidth	REAL	Minimum module width of the code in the image (in pixels). Must be at least 3, preferably 5 - 8.
ipContour	Reference To ITcVnContainer [▶ 385]	Returns the contour of the detected code (ContainerType_Vector_TcVnPoint2_DINT; optional, set to 0 if not required)
eOptions	ULINT	Specifies a combination of one or more ETcVnDataMatrixOptions
nRows	USINT	Specifies the number of code module rows, including the finder and clock pattern, excluding the quiet zone (8 - 144; set to 0 to search for any size)
nCols	USINT	Specifies the number of code module columns, including the finder and clock pattern, excluding the quiet zone (10 - 144; set to 0 to search for any size)
bExtensiveSearch	BOOL	If true, tries different approaches if the specified options fail
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Outputs

Name	Type	Description
fAngleDeg	REAL	Returns the clockwise rotation angle in degree

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Code Reading

Weiterführende Informationen

Die Funktion `F_VN_ReadDataMatrixCodeRoiExp` ist die Expert-Variante von F_VN_ReadDataMatrixCodeRoi [▶ 877] und enthält zusätzliche Parameter.

ipContour

Der Parameter `ipContour` gibt die Code-Eckpunkte als vier Konturpunkte vom Typ `ContainerType_Vector_TcVnPoint2_DINT` zurück. Die Eckpunkte liegen am Rand der äußeren Module. Die Kontur kann z.B. direkt mit F_VN_DrawContours [▶ 1040] gezeichnet oder mit den Funktionen zur Contour Analysis [▶ 985] weiter analysiert werden. Dieser Parameter ist optional und kann auf 0 gesetzt werden, wenn er nicht benötigt wird.

eOptions

Der Parameter `eOptions` definiert, welche Erkennungsstrategien genutzt werden sollen. Dabei existieren drei Gruppen mit sich gegenseitig ausschließenden Optionen, von denen jeweils nur eine Option pro Gruppe ausgewählt werden kann:

1. Bildvorverarbeitung:
   - `TCVN_DMO_BINARY_INPUT`

- TCVN_DMO_GLOBAL_THRESHOLD

2. Gittererkennung:
   - TCVN_DMO_BINARY_GRID
   - TCVN_DMO_STATIC_GRID
3. Code-Farbmodus:
   - TCVN_DMO_REGULAR_COLOR
   - TCVN_DMO_INVERTED_COLOR

Jede Gruppe kann individuell konfiguriert werden, z. B. `TCVN_DMO_GLOBAL_THRESHOLD OR TCVN_DMO_STATIC_GRID OR TCVN_DMO_REGULAR_COLOR`. Ungültige Kombinationen wie `TCVN_DMO_BINARY_GRID OR TCVN_DMO_STATIC_GRID` führen zum Fehler `ADS_E_INVALIDPARM`.

Die Standardeinstellung `TCVN_DMO_DEFAULT` verwendet einen adaptiven lokalen Schwellwert, ein dynamisches Gitter (basierend auf einem Graustufenbild) und erkennt beide Farbmodi. Zur Optimierung der Rechenzeit wird empfohlen, immer den Farbmodus explizit anzugeben.

Eine Beschreibung der Erkennungsstrategien finden Sie unter ETcVnDataMatrixOptions [▶ 237].

nRows und **nCols**

Die Parameter `nRows` und `nCols` legen die erwartete Anzahl der Module in vertikaler (Höhe) bzw. horizontaler (Breite) Richtung fest, bezogen auf einen nicht rotierten Code, bei dem sich der Eckpunkt des L-Patterns unten links befindet. Bei der Anzahl sind sowohl das Finder-Pattern als auch das Clock-Pattern des Data Matrix Codes in die Zählung mit einzubeziehen. Es ist zu empfehlen diese Werte immer zu setzen, um die maximal benötigte Rechenzeit zu reduzieren.

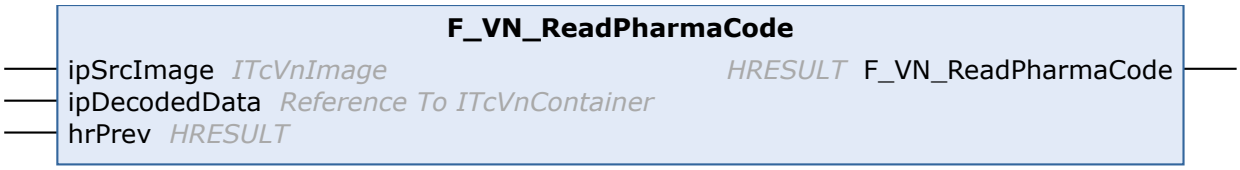
bExtensiveSearch

Der Parameter `bExtensiveSearch` aktiviert eine erweiterte Suchroutine, die bei fehlgeschlagener Erstanalyse automatisch alternative Dekodierungsstrategien anwendet.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.7.9 F_VN_ReadPharmaCode



Detect and interpret a pharma code within the provided image. Can be canceled by Watchdog.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_ReadPharmaCode : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipDecodedData : Reference To ITcVnContainer;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel (all element types) or 3 channel with elements of type TCVN_ET_USINT, TCVN_ET_UINT or TCVN_ET_REAL (3 channel input is expected to be RGB and internally converted to Gray))
ipDecodedData	Reference To ITcVnContainer [▶ 385]	Returns the decoded code (ContainerType_Vector_String_SINT)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Information

Die Funktion `F_VN_ReadPharmaCode` detektiert und liest einen Pharmacode im Eingangsbild `ipSrcImage`. Der Pharmacode ist ein Barcode-Standard. Er ist auf gute Lesbarkeit ausgelegt ist, kann jedoch nur Ganzzahlen im Bereich 3 bis 131070 kodieren. Das Ziel dieses Standards ist eine hohe Lesegeschwindigkeit bei wenig Fehllesungen.

Pharmacodes nicht verwechseln

Verwechseln Sie den Pharmacode Standard nicht mit den Italienischen oder Schweizer Pharmacodes. Diese bezeichnen lediglich Arznei-Identifikationsnummern mit einer Code39-Darstellung.

Aufbau von Pharmacodes

Ein Pharmacode besteht aus dicken und dünnen Balken (dunkel) sowie aus dazwischenliegenden Abständen (hell). Für eine gute Lesbarkeit werden ausreichende Größe und Kontrast des Codes im Bild empfohlen. Die Decodierung geschieht entsprechend folgender Tabelle:

Position	7	6	5	4	3	2	1	0
Dünne Linie	128	64	32	16	8	4	2	1
Dicke Linie	256	128	64	32	16	8	4	2

Detaillierte Informationen zur Spezifikation von Pharma-Codes können Sie online abrufen: Pharma Code Leitfaden von Laetus

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss 1 oder 3 Kanäle haben. 3-kanalige Bilder müssen vom Typ `USINT`, `UINT` oder `REAL` und im Farbraum RGB sein. Sie werden intern in ein Grauwertbild konvertiert.

Gelesene Daten (Rückgabewert)

Die aus dem Code gelesenen Daten werden als String in dem Container `ipDecodedData` zurückgegeben. Der Container ist dabei vom Typ `ContainerType_Vector_String_SINT`.

Expert-Parameter

Die Expert-Variante F_VN_ReadPharmaCodeExp [▶ 882] enthält zusätzliche Parameter.

Daten aus Container lesen

Wurde der Code erfolgreich gefunden und decodiert, wird der Codeinhalt im Container `ipDecodedData` zurückgegeben und kann über die Funktion F_VN_ExportSubContainer_String [▶ 783] in einen String exportiert werden.

```
hr := F_VN_ExportSubContainer_String( ipDecodedData, 0, sText, nMaxLength, hr);
```

Beispiele

- Pharma-Code Reading [▶ 2850]
- Ergebnisauswertung beim Code Reading [▶ 2854]

Verwandte Funktionen

- F_VN_ReadPharmaCode [▶ 880] zum Lesen von Pharma-Codes
- F_VN_ReadBarcode [▶ 865] zum Lesen von Barcodes
- F_VN_ReadDataMatrixCode [▶ 871] zum Lesen von Data-Matrix-Codes
- F_VN_ReadQRCode [▶ 885] zum Lesen von QR-Codes

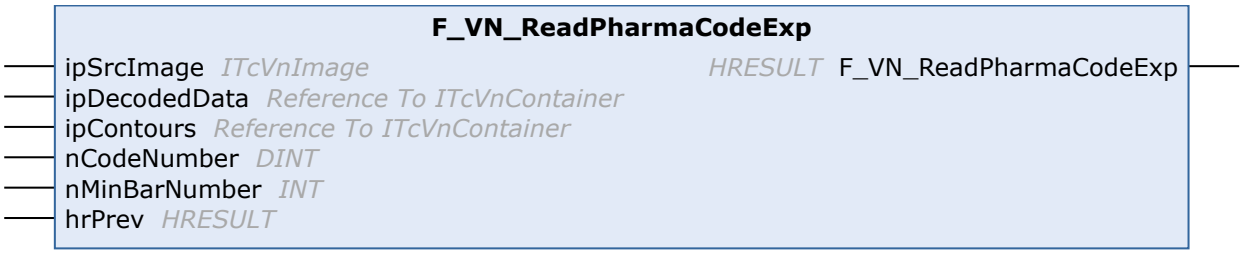
Required License

TC3 Vision Code Reading

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.7.10 F_VN_ReadPharmaCodeExp



Detect and interpret a pharma code within the provided image. Can be canceled by Watchdog.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_ReadPharmaCodeExp : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipDecodedData : Reference To ITcVnContainer;
    ipContours    : Reference To ITcVnContainer;
    nCodeNumber   : DINT;
    nMinBarNumber : INT;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel (all element types) or 3 channel with elements of type TCVN_ET_USINT, TCVN_ET_UINT or TCVN_ET_REAL (3 channel input is expected to be RGB and internally converted to Gray))
ipDecodedData	Reference To ITcVnContainer [▶ 385]	Returns the decoded code (ContainerType_Vector_String_SINT)
ipContours	Reference To ITcVnContainer [▶ 385]	Returns the code positions as contours (optional, set to 0 if not required; ContainerType_Vector_Vector_TcVnPoint2_DINT)
nCodeNumber	DINT	Number of pharma codes that should be detected within the provided image (currently only 1 supported).
nMinBarNumber	INT	Minimal number of (dark) bars that codes must have.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Information

Diese Expertenfunktion erweitert die Funktion F_VN_ReadPharmaCode [▶ 880] wie folgt:

- Rückgabe der Lage von gefundenen Codes über den Parameter `ipContours`
- Angabe der Anzahl der im Bild zu findenden Codes. Sobald `nCodeNumber` oder mehr Codes im Bild gefunden sind, wird die Suche beendet. Wenn weniger als `nCodeNumber` Codes gefunden werden, gibt die Funktion `S_FALSE` zurück.
- Mindestanzahl an dunklen Strichen, die der Code haben muss, um von der Funktion gefunden zu werden, über den Parameter `nMinBarNumber`.

Beispiele

- Pharma-Code Reading [▶ 2850]
- Ergebnisauswertung beim Code Reading [▶ 2854]

Verwandte Funktionen

- F_VN_ReadPharmaCode [▶ 880]
- F_VN_ReadBarcode [▶ 865] bzw. F_VN_ReadBarcodeExp [▶ 868]
- F_VN_ReadDataMatrixCode [▶ 871] bzw. F_VN_ReadDataMatrixCodeExp [▶ 874]
- F_VN_ReadQRCode [▶ 885] bzw. F_VN_ReadQRCodeExp [▶ 888]

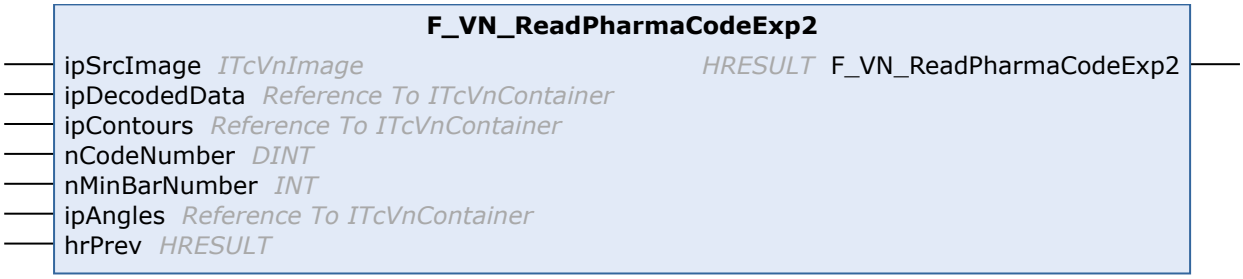
Required License

TC3 Vision Code Reading

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.7.11 F_VN_ReadPharmaCodeExp2



Detect and interpret a pharma code within the provided image. Can be canceled by Watchdog.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_ReadPharmaCodeExp2 : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipDecodedData : Reference To ITcVnContainer;
    ipContours    : Reference To ITcVnContainer;
    nCodeNumber   : DINT;
    nMinBarNumber : INT;
    ipAngles      : Reference To ITcVnContainer;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel (all element types) or 3 channel with elements of type TCVN_ET_USINT, TCVN_ET_UINT or TCVN_ET_REAL (3 channel input is expected to be RGB and internally converted to Gray))
ipDecodedData	Reference To ITcVnContainer [▶ 385]	Returns the decoded code (ContainerType_Vector_String_SINT)
ipContours	Reference To ITcVnContainer [▶ 385]	Returns the code positions as contours (optional, set to 0 if not required; ContainerType_Vector_Vector_TcVnPoint2_DINT)
nCodeNumber	DINT	Number of pharma codes that should be detected within the provided image (currently only 1 supported).
nMinBarNumber	INT	Minimal number of (dark) bars that codes must have.
ipAngles	Reference To ITcVnContainer [▶ 385]	Returns the clockwise code rotation angles (optional, set to 0 if not required; ContainerType_Vector_REAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Information

Diese Expertenfunktion erweitert die Funktion F_VN_ReadPharmaCode [▶ 880] wie folgt:

- Rückgabe der Lage von gefundenen Codes über den Parameter `ipContours`
- Angabe der Anzahl der im Bild zu findenden Codes. Sobald `nCodeNumber` oder mehr Codes im Bild gefunden sind, wird die Suche beendet. Wenn weniger als `nCodeNumber` Codes gefunden werden, gibt die Funktion `S_FALSE` zurück.

- Mindestanzahl an dunklen Strichen, die der Code haben muss, um von der Funktion gefunden zu werden, über den Parameter `nMinBarNumber`.

Drehwinkel

Der Parameter `ipAngles` gibt die ungefähren Drehwinkel der gefundenen Codes im Uhrzeigersinn von -90° bis +90° in Grad zurück. Der Container ist vom Typ `ContainerType_Vector_REAL`. Dieser Parameter ist optional und kann auf 0 gesetzt werden, wenn er nicht benötigt wird.

Unterschiede bei der Leserichtung

Da ein Pharmacode keine Start- oder Stoppkennung beinhaltet, erhält man je nach Leserichtung ein unterschiedliches Ergebnis. Daher kann anhand des Drehwinkels nicht erkannt werden, ob der Code um 180° gedreht ist. Um sicherzustellen, dass der Code in der erwarteten Orientierung vorliegt, muss das Bild um 180° gedreht und der Code erneut gelesen werden. Dann können beide gelesenen Ergebnisse mit einem erwarteten Wert verglichen werden, um so die Orientierung eindeutig bestimmen zu können.

Beispiele

- Pharma-Code Reading [▶ 2850]
- Ergebnisauswertung beim Code Reading [▶ 2854]

Verwandte Funktionen

- F_VN_ReadPharmaCode [▶ 880]
- F_VN_ReadBarcode [▶ 865] bzw. F_VN_ReadBarcodeExp [▶ 868]
- F_VN_ReadDataMatrixCode [▶ 871] bzw. F_VN_ReadDataMatrixCodeExp [▶ 874]
- F_VN_ReadQRCode [▶ 885] bzw. F_VN_ReadQRCodeExp [▶ 888]

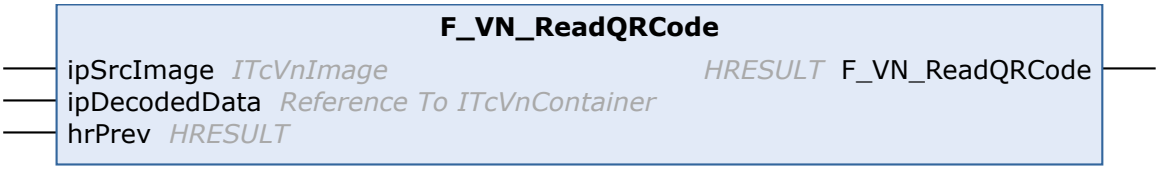
Required License

TC3 Vision Code Reading

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.7.12 F_VN_ReadQRCode



Detect and interpret a QR code within the provided image. Can be canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_ReadQRCode : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipDecodedData : Reference To ITcVnContainer;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT elements, 1 channel or 3 channel (3 channel input is expected to be RGB and internally converted to Gray))
ipDecodedData	Reference To ITcVnContainer [▶ 385]	Returns the decoded code (ContainerType_Vector_String_SINT)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Der QR-Code (Quick-Response-Code) nach IEC 18004 ist ein 2D-Code, welcher in Deutschland besonders im Bereich des Mobile Tagging verbreitet ist. Durch die Möglichkeit Kanji/Kana-Zeichen zu codieren, hat sich der QR-Code im asiatischen Raum auch in der Industrie etabliert.

Die Funktion F_VN_ReadQRCode kann diesen Codetyp im Bild erkennen und dekodieren. Dabei wird von einem Code pro Bild ausgegangen.

QR-Code Aufbau

1.		Positionsmarkierungen
2.		Ausrichtungsmarkierung
3.		Synchronisationsreihen
4.		Versionsinformationen
5.		Datenformat
6.		Datenfeld
7.		Ruhezone, mind. 4 Zellen breit

Störungen / Beschädigungen können eine Detektion und Dekodierung verhindern.

Codesuche

Die Funktion F_VN_ReadQRCode sucht von oben links nach unten rechts zeilenweise nach einem QR Code im Bild.

Daten aus Container lesen

Wurde der Code erfolgreich gefunden und decodiert, wird der Codeinhalt im Container `ipDecodedData` zurückgegeben und kann über die Funktion F_VN_ExportSubContainer_String [▶ 783] in einen String exportiert werden.

```
hr := F_VN_ExportSubContainer_String( ipDecodedData, 0, sText, nMaxLength, hr);
```

HRESULT

Wie alle TwinCAT Vision API-Funktionen gibt auch diese Funktion ein HRESULT [▶ 150] zurück, um anzugeben, ob die Ausführung erfolgreich war oder fehlgeschlagen ist. Im Erfolgsfall kann bei der nachfolgenden Ergebnisbehandlung zwischen folgenden Success-Codes unterschieden werden. Um zwischen diesen Fällen zu unterscheiden, kann die `HRESULT`-Variable direkt mit `S_OK` oder `S_FALSE` verglichen werden.

Code	Name	Beschreibung
16#000	S_OK	Die Funktion wurde erfolgreich ausgeführt. Die angegebene Anzahl an Codes wurde im Bild gefunden.
16#001	S_FALSE	Die Funktion wurde erfolgreich ausgeführt. Es wurden weniger Codes als angegeben im Bild gefunden.
16#256	S_WATCHDOG TIMEOUT	Die Funktion wurde vom Watchdog [▶ 156] abgebrochen.

Beispiele

- QR-Code Reading [▶ 2852]
- Ergebnisauswertung beim Code Reading [▶ 2854]

Verwandte Funktionen

- F_VN_ReadQRCodeExp [▶ 888]
- F_VN_ReadBarcode [▶ 865] bzw. F_VN_ReadBarcodeExp [▶ 867]
- F_VN_ReadDataMatrixCode [▶ 871] bzw. F_VN_ReadDataMatrixCodeExp [▶ 874]
- F_VN_ReadPharmaCode [▶ 880] bzw. F_VN_ReadPharmaCodeExp [▶ 882]

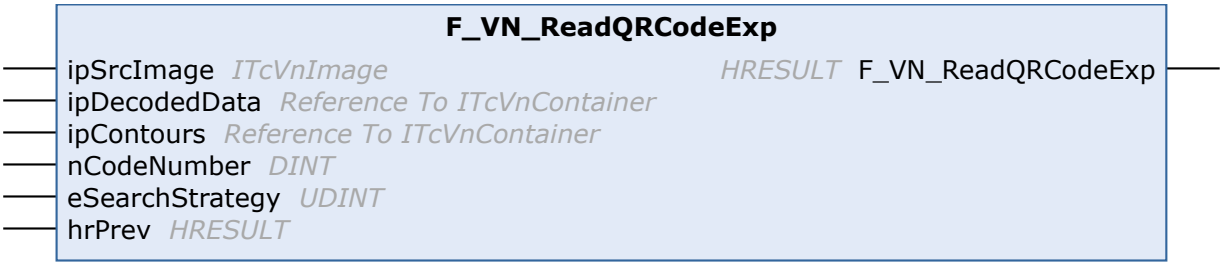
Required License

TC3 Vision Code Reading

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.7.13 F_VN_ReadQRCodeExp



Detect and interpret a QR code within the provided image. Can be canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_ReadQRCodeExp : HRESULT
VAR_INPUT
    ipSrcImage      : ITcVnImage;
    ipDecodedData   : Reference To ITcVnContainer;
    ipContours      : Reference To ITcVnContainer;
    nCodeNumber     : DINT;
    eSearchStrategy : UDINT;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT elements, 1 channel or 3 channel (3 channel input is expected to be RGB and internally converted to Gray))
ipDecodedData	Reference To ITcVnContainer [▶ 385]	Returns the decoded code (ContainerType_Vector_String_SINT)
ipContours	Reference To ITcVnContainer [▶ 385]	Returns the code positions as contours (optional, set to 0 if not required; ContainerType_Vector_Vector_TcVnPoint2_DINT)
nCodeNumber	DINT	Number of QR codes that should be detected within the provided image (currently only 1 supported).
eSearchStrategy	UDINT	Used search strategy (ETcVn2dCodeSearchStrategy)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

HINWEIS

Floating Point Exceptions

Diese Funktion kann Fehler herbeiführen, wenn die Option **Floating point exceptions** der ausführenden SPS-Task [▶ 63] aktiv ist. Deaktivieren Sie diese Option deshalb.

Die Funktion F_VN_ReadQRCodeExp entspricht der Funktion F_VN_ReadQRCode [▶ 885] erweitert um die folgenden Parameter.

ipContour

Der Parameter `ipContour` gibt die gefundene Codeposition als Kontur zurück.

nCodeNumber

Der Parameter `nCodeNumber` definiert, nach wie vielen Codes gesucht werden soll.

> **Suche nach mehreren QR-Codes aktuell nicht unterstützt**
>
> Die Suche nach mehreren QR-Codes in einem Bild wird aktuell nicht unterstützt. Daher ist der Parameter nCodeNumber := 1 zu setzen.

eSearchStrategy

Der Parameter `eSearchStrategy` definiert, welche Suchstrategien für die Codesuche genutzt werden sollen. Die Standard-Option `TCVN_CSS_DEFAULT` führt zu den Suchstrategien `TCVN_CSS_FIRST_NOT_INVERTED` und `TCVN_CSS_FIRST_NOT_FLIPPED`. Eine Beschreibung der Suchstrategien finden Sie unter Code Reading Suchstrategien [▶ 864].

Beispiele

- QR-Code Reading [▶ 2852]
- Ergebnisauswertung beim Code Reading [▶ 2854]

Verwandte Funktionen

- F_VN_ReadQRCode [▶ 885]
- F_VN_ReadBarcode [▶ 865] bzw. F_VN_ReadBarcodeExp [▶ 867]
- F_VN_ReadDataMatrixCode [▶ 871] bzw. F_VN_ReadDataMatrixCodeExp [▶ 874]
- F_VN_ReadPharmaCode [▶ 880] bzw. F_VN_ReadPharmaCodeExp [▶ 882]

Required License

TC3 Vision Code Reading

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.7.14 F_VN_ReadQRCodeExp2

Detect and interpret a QR code within the provided image. Can be canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_ReadQRCodeExp2 : HRESULT
VAR_INPUT
    ipSrcImage      : ITcVnImage;
    ipDecodedData   : Reference To ITcVnContainer;
    ipContours      : Reference To ITcVnContainer;
    nCodeNumber     : DINT;
    eSearchStrategy : UDINT;
```

```
    ipAngles        : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT elements, 1 channel or 3 channel (3 channel input is expected to be RGB and internally converted to Gray))
ipDecodedData	Reference To ITcVnContainer [▶ 385]	Returns the decoded code (ContainerType_Vector_String_SINT)
ipContours	Reference To ITcVnContainer [▶ 385]	Returns the code positions as contours (optional, set to 0 if not required; ContainerType_Vector_Vector_TcVnPoint2_DINT)
nCodeNumber	DINT	Number of QR codes that should be detected within the provided image (currently only 1 supported).
eSearchStrategy	UDINT	Used search strategy (ETcVn2dCodeSearchStrategy)
ipAngles	Reference To ITcVnContainer [▶ 385]	Returns the clockwise code rotation angles (optional, set to 0 if not required; ContainerType_Vector_REAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

HINWEIS

Floating Point Exceptions

Diese Funktion kann Fehler herbeiführen, wenn die Option **Floating point exceptions** der ausführenden SPS-Task [▶ 63] aktiv ist. Deaktivieren Sie diese Option deshalb.

Die Funktion F_VN_ReadQRCodeExp entspricht der Funktion F_VN_ReadQRCode [▶ 885] erweitert um die folgenden Parameter.

ipContour

Der Parameter `ipContour` gibt die gefundene Codeposition als Kontur zurück.

nCodeNumber

Der Parameter `nCodeNumber` definiert, nach wie vielen Codes gesucht werden soll.

Suche nach mehreren QR-Codes aktuell nicht unterstützt

Die Suche nach mehreren QR-Codes in einem Bild wird aktuell nicht unterstützt. Daher ist der Parameter nCodeNumber := 1 zu setzen.

eSearchStrategy

Der Parameter `eSearchStrategy` definiert, welche Suchstrategien für die Codesuche genutzt werden sollen. Die Standard-Option `TCVN_CSS_DEFAULT` führt zu den Suchstrategien `TCVN_CSS_FIRST_NOT_INVERTED` und `TCVN_CSS_FIRST_NOT_FLIPPED`. Eine Beschreibung der Suchstrategien finden Sie unter Code Reading Suchstrategien [▶ 864].

Drehwinkel

Der Parameter `ipAngles` gibt die Drehwinkel der gefundenen Codes im Uhrzeigersinn in Grad zurück. Der Container ist vom Typ `ContainerType_Vector_REAL`. Dieser Parameter ist optional und kann auf 0 gesetzt werden, wenn er nicht benötigt wird.

Beispiele

- QR-Code Reading [▶ 2852]
- Ergebnisauswertung beim Code Reading [▶ 2854]

Verwandte Funktionen

- F_VN_ReadQRCode [▶ 885]
- F_VN_ReadBarcode [▶ 865] bzw. F_VN_ReadBarcodeExp [▶ 867]
- F_VN_ReadDataMatrixCode [▶ 871] bzw. F_VN_ReadDataMatrixCodeExp [▶ 874]
- F_VN_ReadPharmaCode [▶ 880] bzw. F_VN_ReadPharmaCodeExp [▶ 882]

Required License

TC3 Vision Code Reading

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8 Container Statistics

Diese Gruppe enthält Funktionen zur statistischen Analyse von Containern [▶ 163].

Funktionen

Mit euklidischer Norm

- F_VN_ContainerAverage [▶ 973]
- F_VN_ContainerAverageVariance [▶ 977]
- F_VN_MaxElement [▶ 982]
- F_VN_MedianElement [▶ 983]
- F_VN_MinElement [▶ 984]

Elementweise

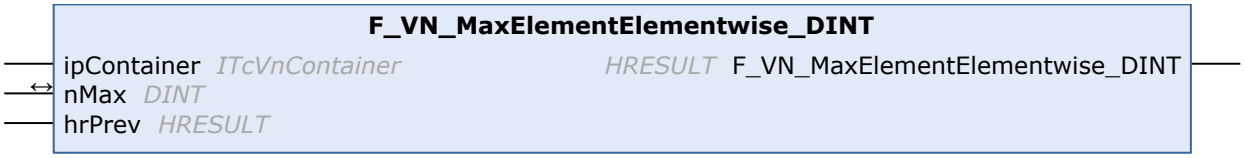
- F_VN_ContainerAverageElementwise [▶ 974]
- F_VN_ContainerAverageVarianceElementwise [▶ 978]
- F_VN_MaxElementElementwise [▶ 891]
- F_VN_MedianElementElementwise [▶ 919]
- F_VN_MinElementElementwise [▶ 945]

6.1.4.8.1 F_VN_MaxElementElementwise

Dieses Kapitel enthält Funktionen, mit denen der Max-Wert entsprechend dem Datentyp Elementweise ermittelt wird. Diese Funktion ist für Container mit Grundelementen verfügbar.

Z. B. bei einem TcVnPoint2_DINT werden der maximale X-Wert und der maximale Y-Wert unabhängig voneinander ermittelt, daher müssen die Werte nicht von dem gleichen Punkt stammen.

6.1.4.8.1.1 F_VN_MaxElementElementwise_DINT



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nMax        : DINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nMax	DINT	Returns the requested element

Return value

HRESULT [▶ 150]

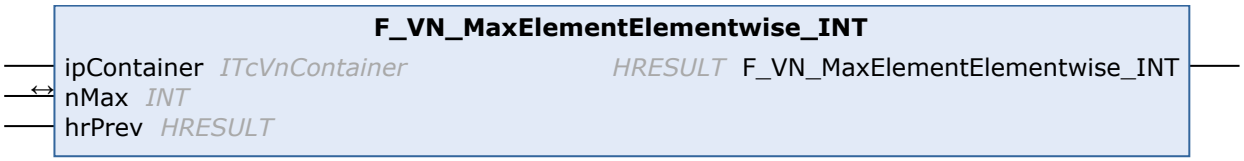
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.2 F_VN_MaxElementElementwise_INT



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nMax        : INT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nMax	INT	Returns the requested element

Return value

HRESULT [▶ 150]

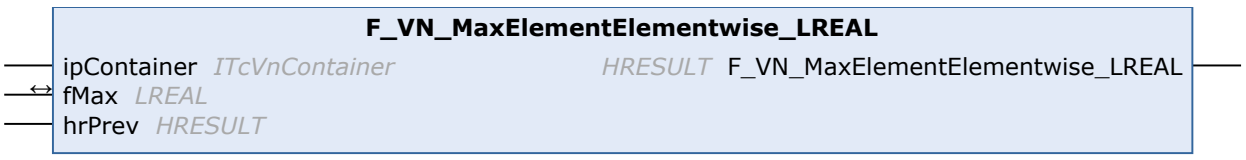
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.3 F_VN_MaxElementElementwise_LREAL



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    fMax        : LREAL;
END_VAR
```

```
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fMax	LREAL	Returns the requested element

Return value

HRESULT [▶ 150]

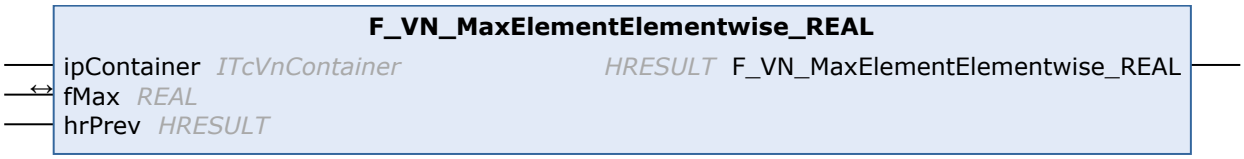
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.4 F_VN_MaxElementElementwise_REAL



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    fMax        : REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fMax	REAL	Returns the requested element

Return value

HRESULT [▶ 150]

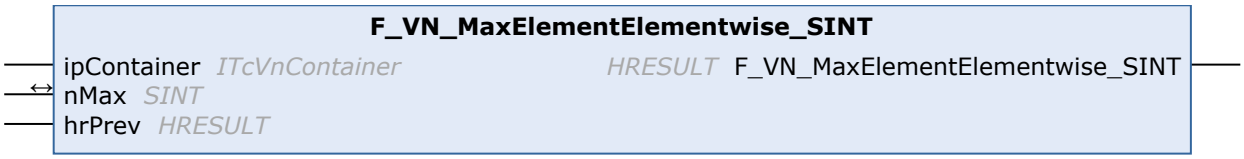
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.5 F_VN_MaxElementElementwise_SINT



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nMax        : SINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nMax	SINT	Returns the requested element

Return value

HRESULT [▶ 150]

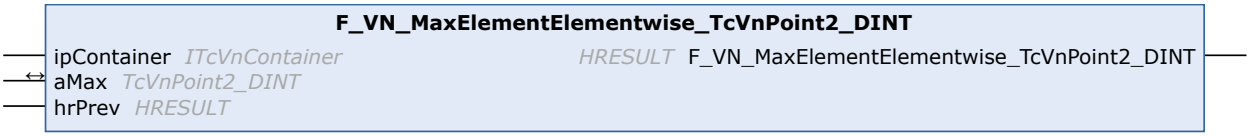
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.6 F_VN_MaxElementElementwise_TcVnPoint2_DINT



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnPoint2_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnPoint2_DINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnPoint2_DINT [▶ 171]	Returns the requested element

Return value

HRESULT [▶ 150]

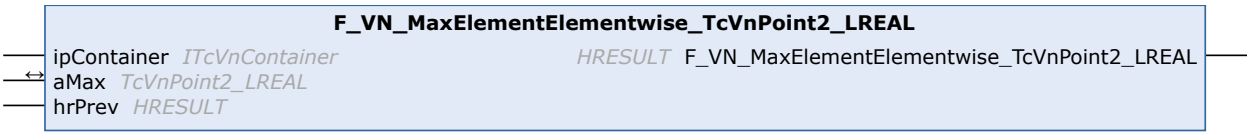
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.7 F_VN_MaxElementElementwise_TcVnPoint2_LREAL



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnPoint2_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnPoint2_LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnPoint2_LREAL [▶ 171]	Returns the requested element

Return value

HRESULT [▶ 150]

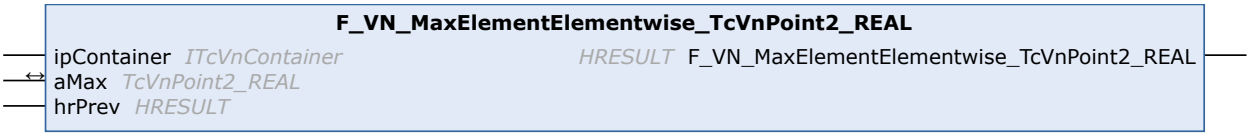
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.8 F_VN_MaxElementElementwise_TcVnPoint2_REAL



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnPoint2_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnPoint2_REAL [▶ 171]	Returns the requested element

Return value

HRESULT [▶ 150]

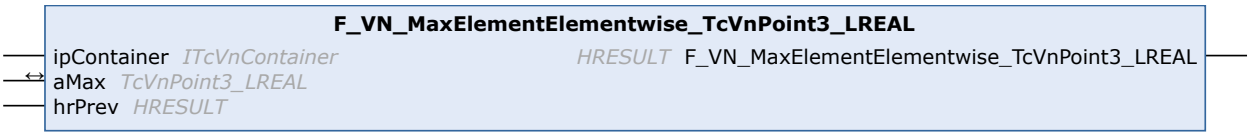
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.9 F_VN_MaxElementElementwise_TcVnPoint3_LREAL



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnPoint3_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnPoint3_LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnPoint3_LREAL [▶ 171]	Returns the requested element

Return value

HRESULT [▶ 150]

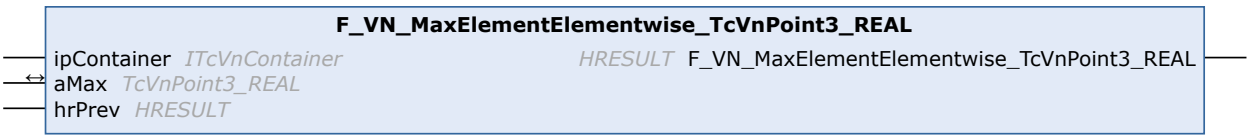
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.10 F_VN_MaxElementElementwise_TcVnPoint3_REAL



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnPoint3_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnPoint3_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnPoint3_REAL [▶ 171]	Returns the requested element

Return value

HRESULT [▶ 150]

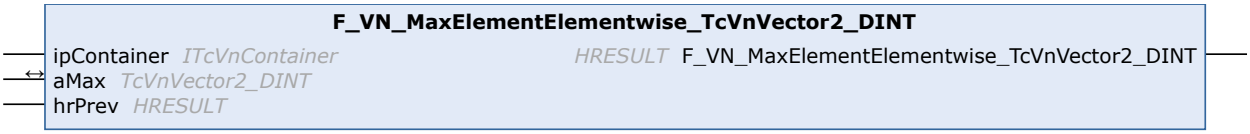
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.11 F_VN_MaxElementElementwise_TcVnVector2_DINT



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector2_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnVector2_DINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector2_DINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

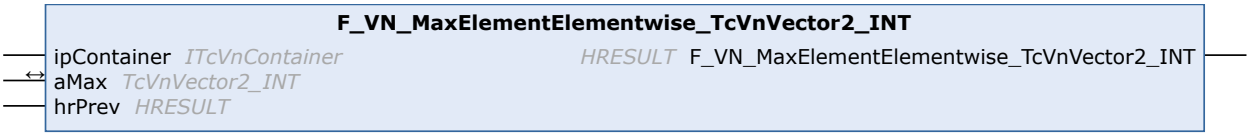
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.12 F_VN_MaxElementElementwise_TcVnVector2_INT



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector2_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnVector2_INT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector2_INT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

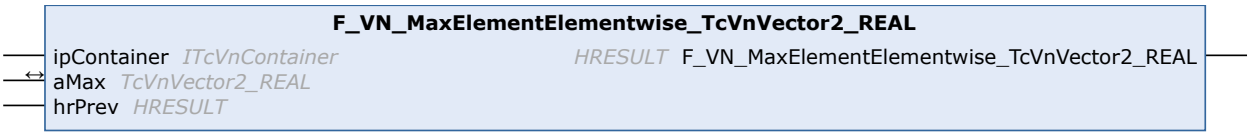
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.13 F_VN_MaxElementElementwise_TcVnVector2_REAL



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector2_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnVector2_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector2_REAL [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

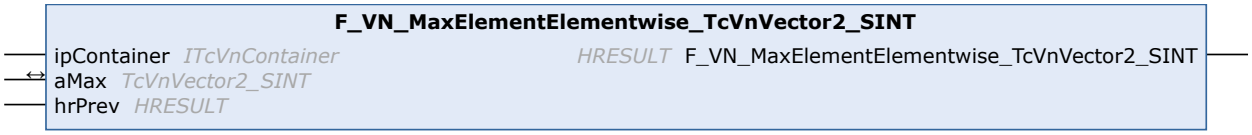
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.14 F_VN_MaxElementElementwise_TcVnVector2_SINT



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector2_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnVector2_SINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector2_SINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

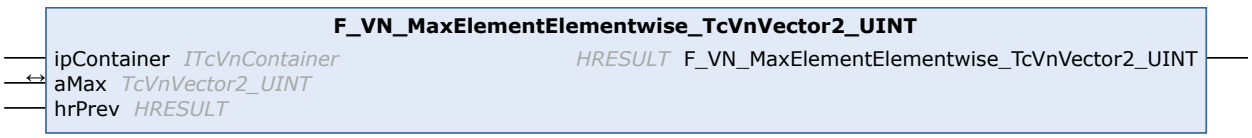
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.15 F_VN_MaxElementElementwise_TcVnVector2_UINT



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector2_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnVector2_UINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector2_UINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

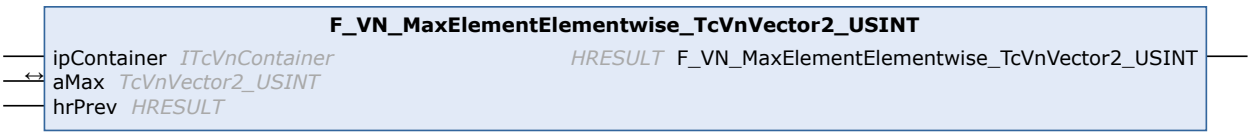
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.16 F_VN_MaxElementElementwise_TcVnVector2_USINT



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector2_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnVector2_USINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector2_USINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

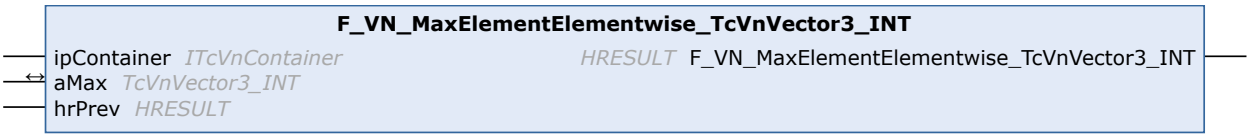
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.17 F_VN_MaxElementElementwise_TcVnVector3_INT



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector3_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnVector3_INT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector3_INT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

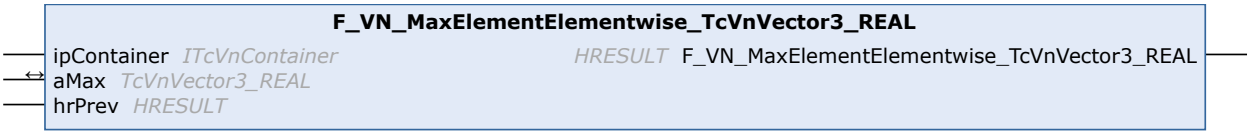
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.18 F_VN_MaxElementElementwise_TcVnVector3_REAL



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector3_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnVector3_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector3_REAL [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

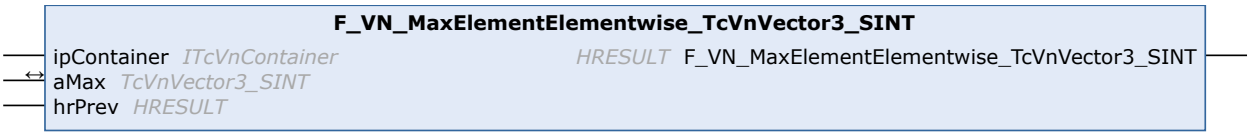
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.19 F_VN_MaxElementElementwise_TcVnVector3_SINT



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector3_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnVector3_SINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector3_SINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

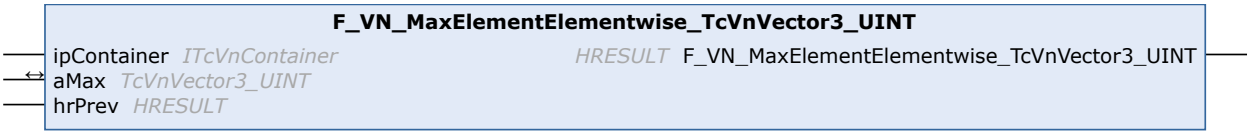
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.20 F_VN_MaxElementElementwise_TcVnVector3_UINT



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector3_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnVector3_UINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector3_UINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

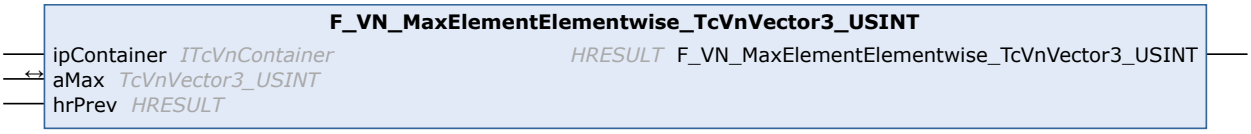
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.21 F_VN_MaxElementElementwise_TcVnVector3_USINT



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector3_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnVector3_USINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector3_USINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

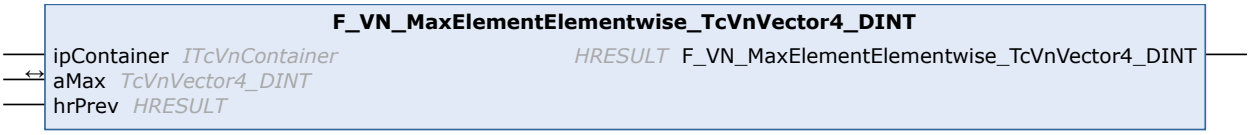
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.22 F_VN_MaxElementElementwise_TcVnVector4_DINT



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector4_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnVector4_DINT;
END_VAR
```

```
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector4_DINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

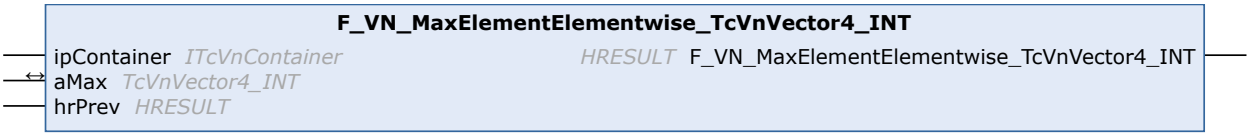
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.23 F_VN_MaxElementElementwise_TcVnVector4_INT



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector4_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnVector4_INT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector4_INT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

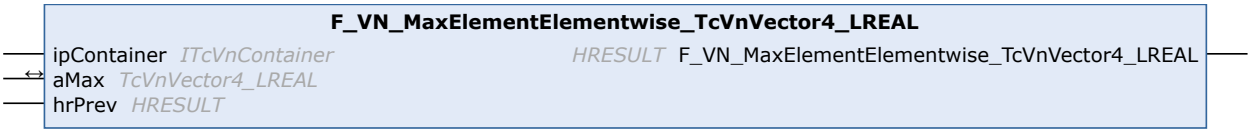
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.24 F_VN_MaxElementElementwise_TcVnVector4_LREAL



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector4_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector4_LREAL [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

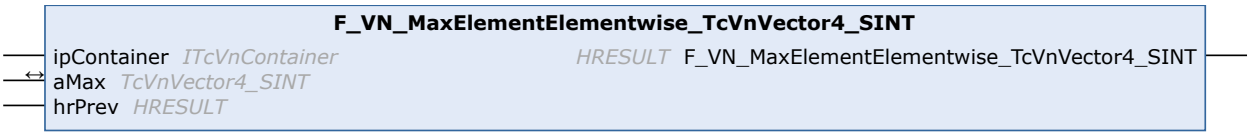
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.25 F_VN_MaxElementElementwise_TcVnVector4_SINT



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector4_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnVector4_SINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector4_SINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

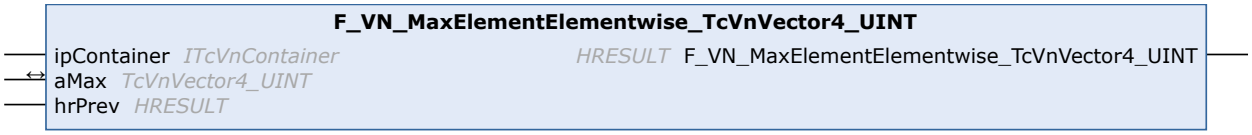
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.26 F_VN_MaxElementElementwise_TcVnVector4_UINT



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector4_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnVector4_UINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector4_UINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

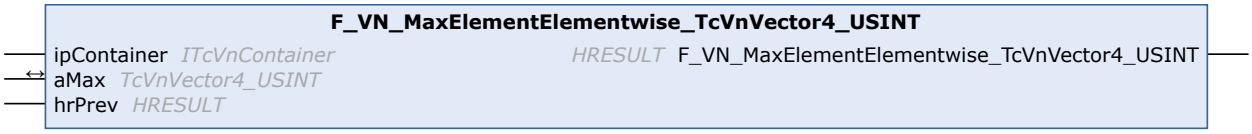
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.27 F_VN_MaxElementElementwise_TcVnVector4_USINT



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector4_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnVector4_USINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector4_USINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

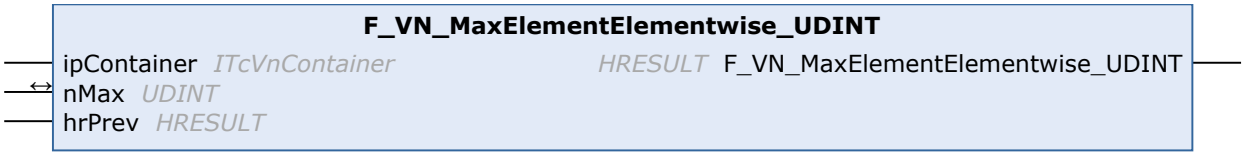
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.28 F_VN_MaxElementElementwise_UDINT



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_UDINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nMax        : UDINT;
END_VAR
```

```
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nMax	UDINT	Returns the requested element

Return value

HRESULT [▶ 150]

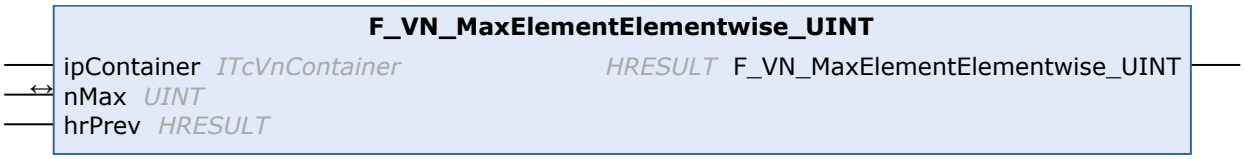
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.29 F_VN_MaxElementElementwise_UINT



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nMax        : UINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nMax	UINT	Returns the requested element

Return value

HRESULT [▶ 150]

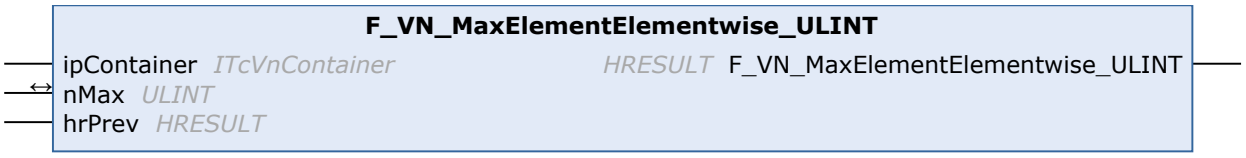
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.30 F_VN_MaxElementElementwise_ULINT



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_ULINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nMax        : ULINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nMax	ULINT	Returns the requested element

Return value

HRESULT [▶ 150]

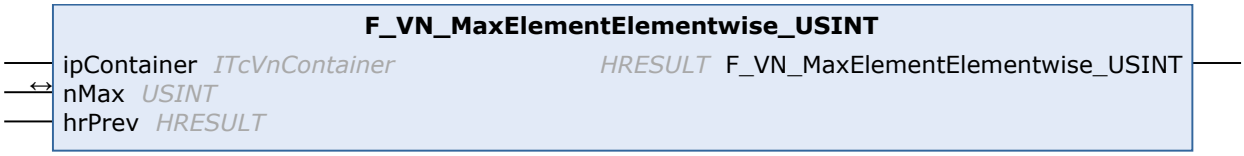
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.31 F_VN_MaxElementElementwise_USINT



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nMax        : USINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nMax	USINT	Returns the requested element

Return value

HRESULT [▶ 150]

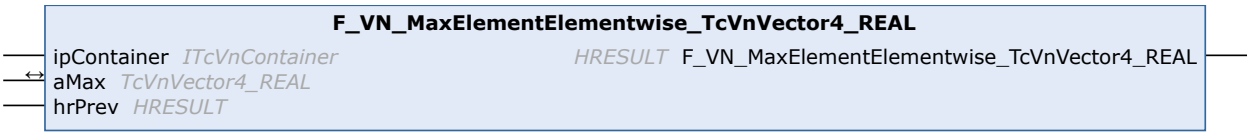
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.1.32 F_VN_MaxElementElementwise_TcVnVector4_REAL



Gets the element wise maximum container element.

Syntax

Definition:

```
FUNCTION F_VN_MaxElementElementwise_TcVnVector4_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMax        : TcVnVector4_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMax	TcVnVector4_REAL [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

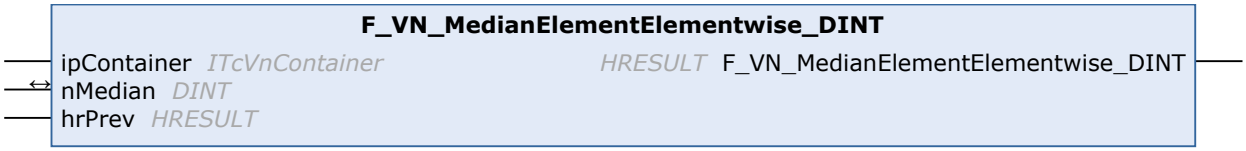
Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2 F_VN_MedianElementElementwise

Dieses Kapitel enthält Funktionen, mit denen der Median-Wert entsprechend dem Datentyp Elementweise ermittelt wird. Diese Funktion ist für Container mit Grundelementen verfügbar.

Z. B. bei einem TcVnPoint2_DINT werden der median-X-Wert und der median-Y-Wert unabhängig voneinander ermittelt, daher müssen die Werte nicht von dem gleichen Punkt stammen.

6.1.4.8.2.1 F_VN_MedianElementElementwise_DINT



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nMedian     : DINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nMedian	DINT	Returns the requested element

Return value

HRESULT [▶ 150]

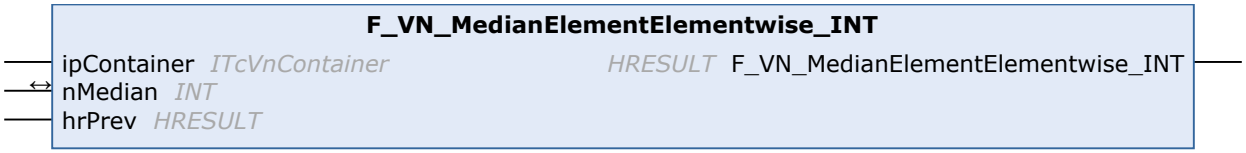
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.2 F_VN_MedianElementElementwise_INT



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nMedian     : INT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nMedian	INT	Returns the requested element

Return value

HRESULT [▶ 150]

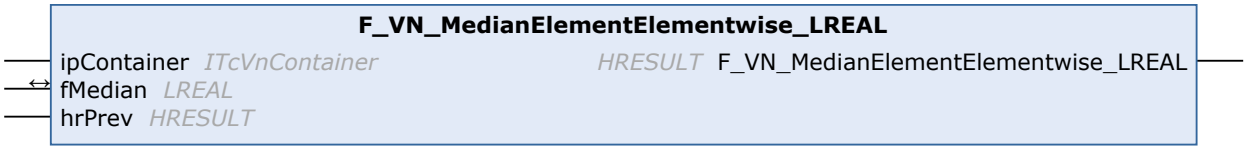
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.3 F_VN_MedianElementElementwise_LREAL



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    fMedian     : LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fMedian	LREAL	Returns the requested element

Return value

HRESULT [▶ 150]

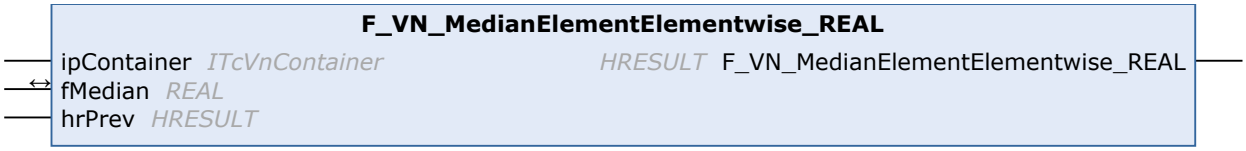
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.4 F_VN_MedianElementElementwise_REAL



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    fMedian     : REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fMedian	REAL	Returns the requested element

Return value

HRESULT [▶ 150]

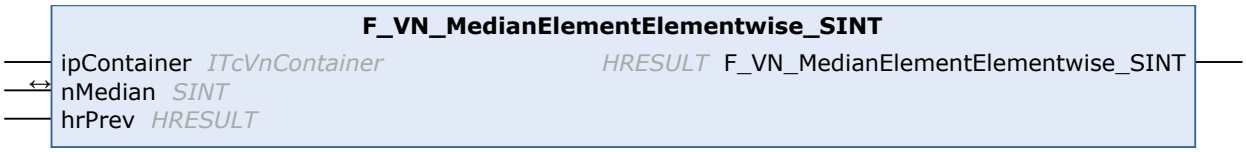
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.5 F_VN_MedianElementElementwise_SINT



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nMedian     : SINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nMedian	SINT	Returns the requested element

Return value

HRESULT [▶ 150]

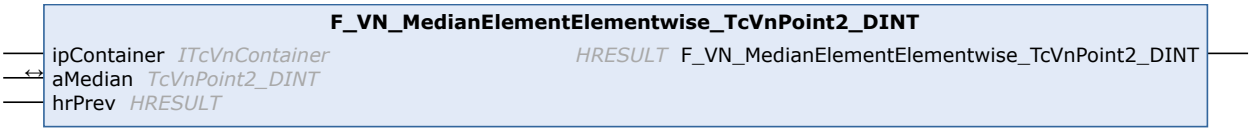
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.6 F_VN_MedianElementElementwise_TcVnPoint2_DINT



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnPoint2_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnPoint2_DINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnPoint2_DINT [▶ 171]	Returns the requested element

Return value

HRESULT [▶ 150]

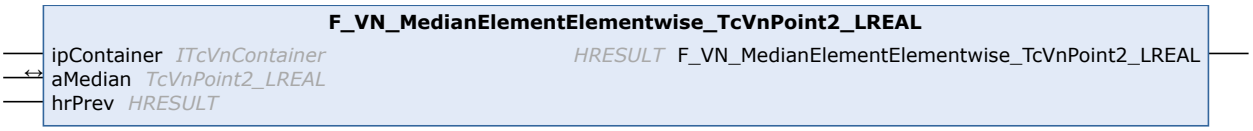
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.7 F_VN_MedianElementElementwise_TcVnPoint2_LREAL



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnPoint2_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnPoint2_LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnPoint2_LREAL [▶ 171]	Returns the requested element

Return value

HRESULT [▶ 150]

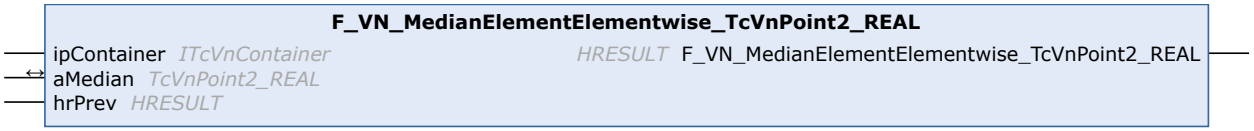
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.8 F_VN_MedianElementElementwise_TcVnPoint2_REAL



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnPoint2_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnPoint2_REAL [▶ 171]	Returns the requested element

Return value

HRESULT [▶ 150]

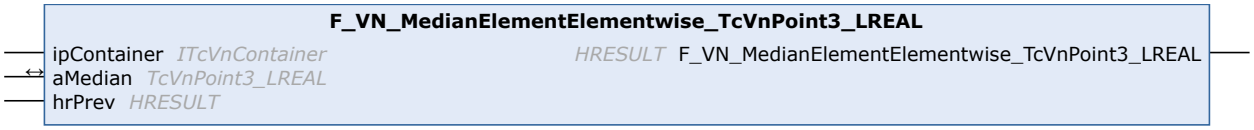
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.9 F_VN_MedianElementElementwise_TcVnPoint3_LREAL



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnPoint3_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnPoint3_LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnPoint3_LREAL [▶ 171]	Returns the requested element

Return value

HRESULT [▶ 150]

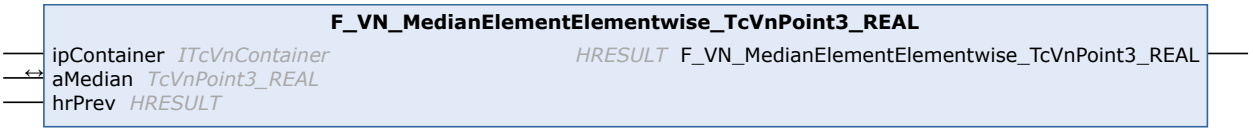
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.10 F_VN_MedianElementElementwise_TcVnPoint3_REAL



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnPoint3_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnPoint3_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnPoint3_REAL [▶ 171]	Returns the requested element

Return value

HRESULT [▶ 150]

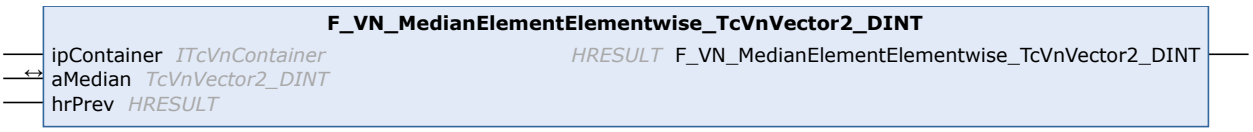
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.11 F_VN_MedianElementElementwise_TcVnVector2_DINT



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnVector2_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnVector2_DINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector2_DINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

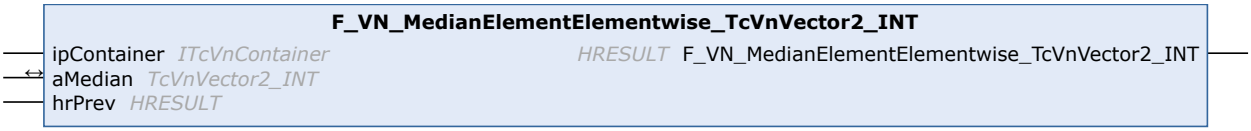
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.12 F_VN_MedianElementElementwise_TcVnVector2_INT



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnVector2_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnVector2_INT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector2_INT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

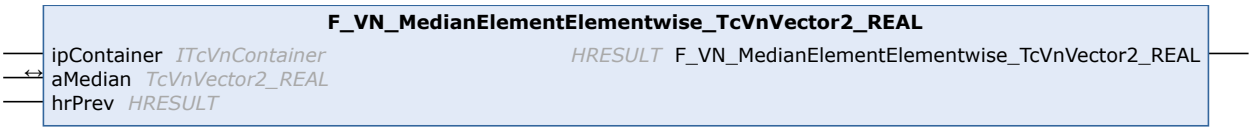
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.13 F_VN_MedianElementElementwise_TcVnVector2_REAL



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnVector2_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnVector2_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector2_REAL [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

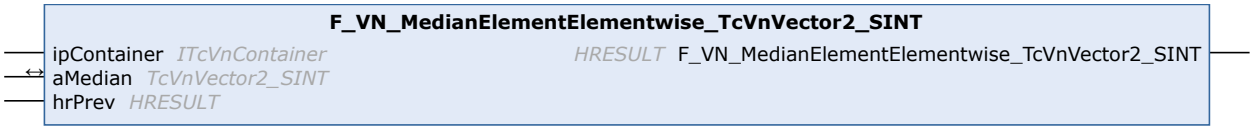
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.14 F_VN_MedianElementElementwise_TcVnVector2_SINT



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnVector2_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnVector2_SINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector2_SINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

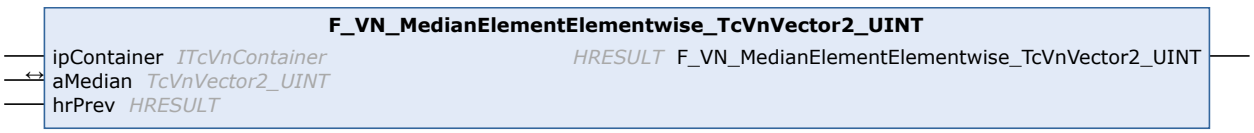
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.15 F_VN_MedianElementElementwise_TcVnVector2_UINT



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnVector2_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnVector2_UINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector2_UINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

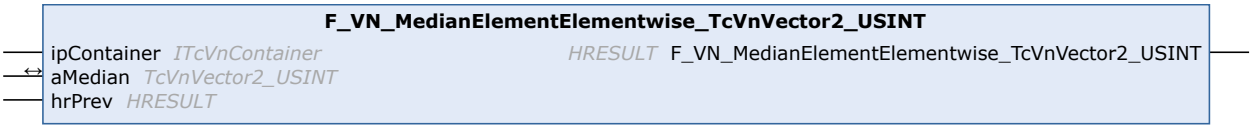
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.16 F_VN_MedianElementElementwise_TcVnVector2_USINT



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnVector2_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnVector2_USINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector2_USINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

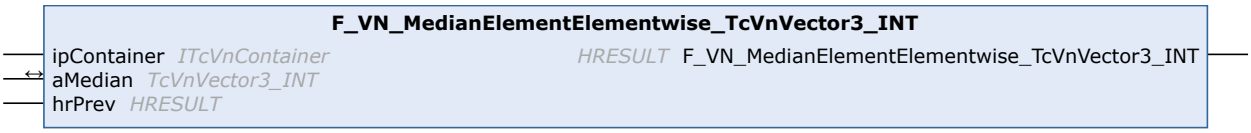
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.17 F_VN_MedianElementElementwise_TcVnVector3_INT



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnVector3_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnVector3_INT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector3_INT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

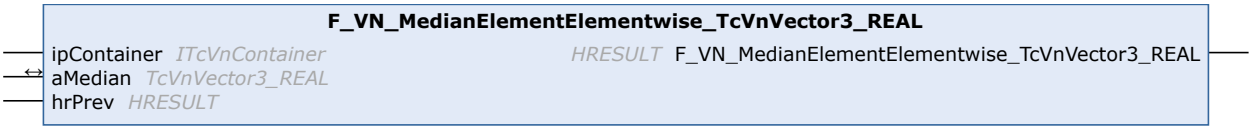
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.18 F_VN_MedianElementElementwise_TcVnVector3_REAL



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnVector3_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnVector3_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector3_REAL [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

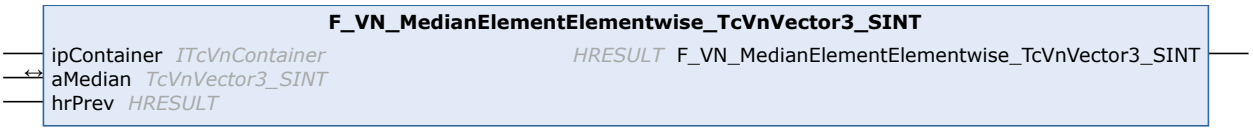
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.19 F_VN_MedianElementElementwise_TcVnVector3_SINT



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnVector3_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnVector3_SINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector3_SINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

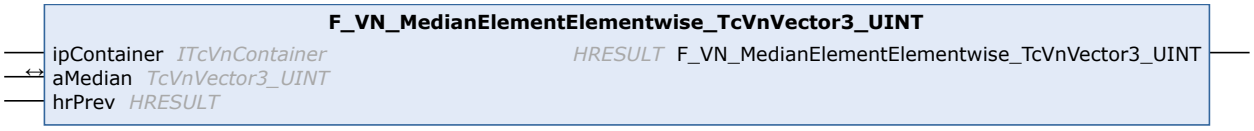
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.20 F_VN_MedianElementElementwise_TcVnVector3_UINT



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnVector3_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnVector3_UINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector3_UINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

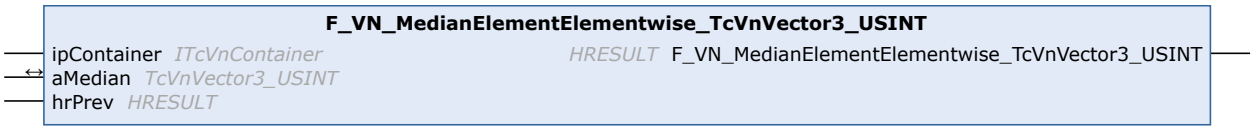
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.21 F_VN_MedianElementElementwise_TcVnVector3_USINT



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnVector3_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnVector3_USINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector3_USINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

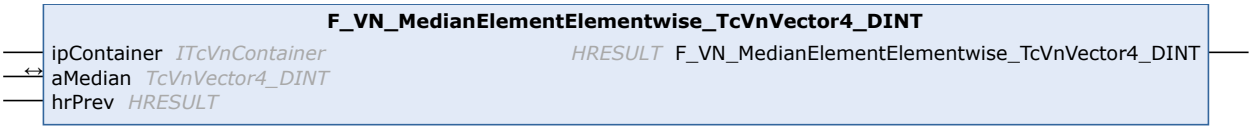
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.22 F_VN_MedianElementElementwise_TcVnVector4_DINT



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnVector4_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnVector4_DINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector4_DINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

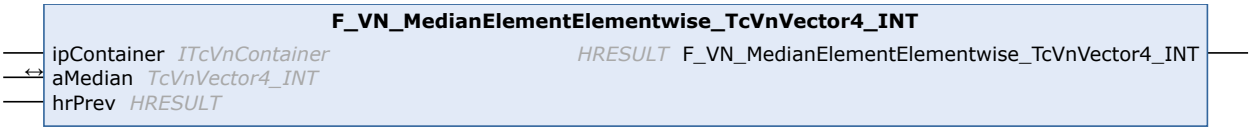
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.23 F_VN_MedianElementElementwise_TcVnVector4_INT



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnVector4_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnVector4_INT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector4_INT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

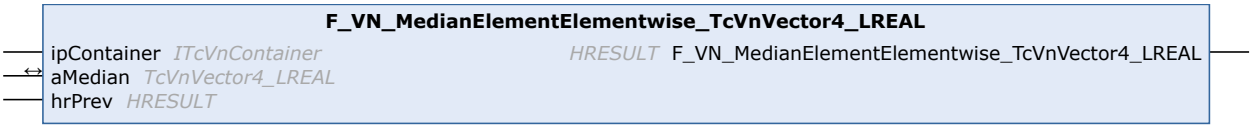
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.24 F_VN_MedianElementElementwise_TcVnVector4_LREAL



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnVector4_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector4_LREAL [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

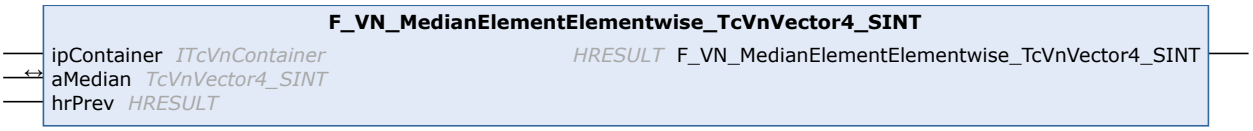
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.25 F_VN_MedianElementElementwise_TcVnVector4_SINT



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnVector4_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnVector4_SINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector4_SINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

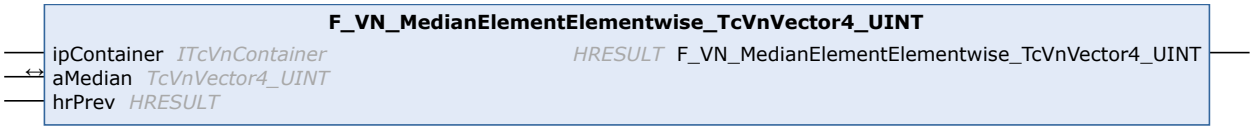
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.26 F_VN_MedianElementElementwise_TcVnVector4_UINT



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnVector4_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnVector4_UINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector4_UINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

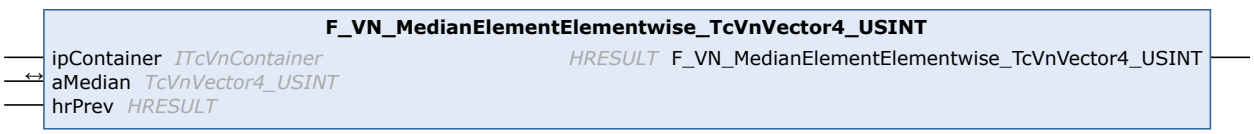
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.27 F_VN_MedianElementElementwise_TcVnVector4_USINT



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnVector4_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnVector4_USINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector4_USINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

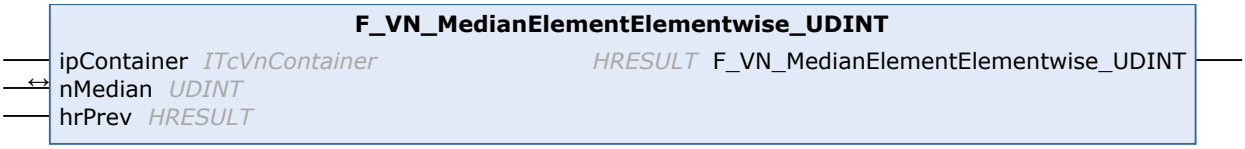
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.28 F_VN_MedianElementElementwise_UDINT



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_UDINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nMedian     : UDINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nMedian	UDINT	Returns the requested element

Return value

HRESULT [▶ 150]

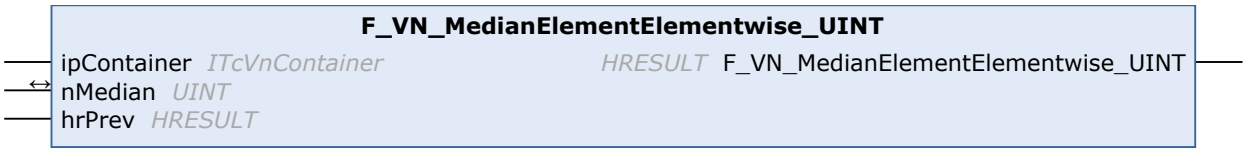
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.29 F_VN_MedianElementElementwise_UINT



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nMedian     : UINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nMedian	UINT	Returns the requested element

Return value

HRESULT [▶ 150]

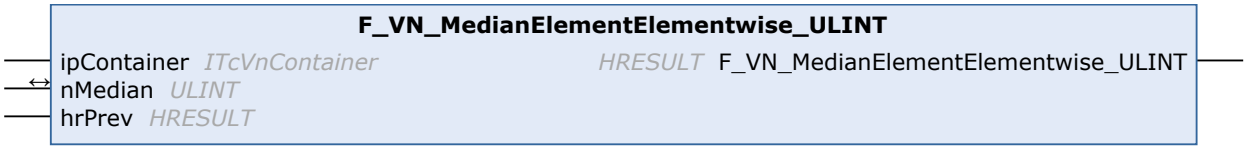
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.30 F_VN_MedianElementElementwise_ULINT



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_ULINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nMedian     : ULINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nMedian	ULINT	Returns the requested element

Return value

HRESULT [▶ 150]

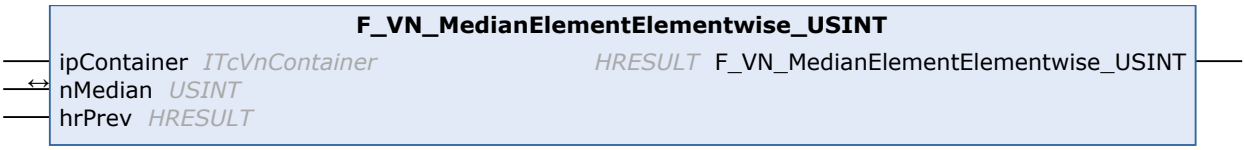
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.31 F_VN_MedianElementElementwise_USINT



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nMedian     : USINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nMedian	USINT	Returns the requested element

Return value

HRESULT [▶ 150]

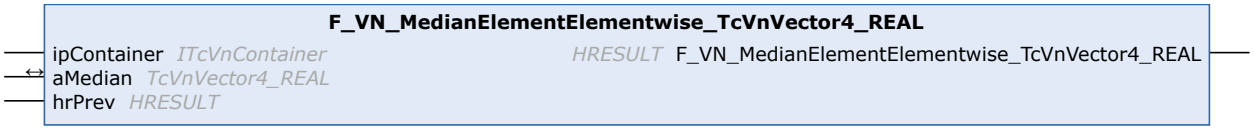
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.2.32 F_VN_MedianElementElementwise_TcVnVector4_REAL



Gets the element wise median container element.

Syntax

Definition:

```
FUNCTION F_VN_MedianElementElementwise_TcVnVector4_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMedian     : TcVnVector4_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector4_REAL [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

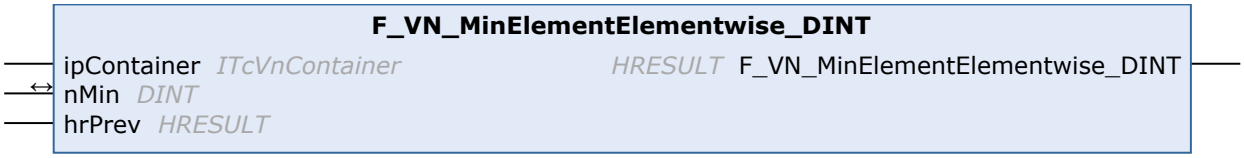
Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3 F_VN_MinElementElementwise

Dieses Kapitel enthält Funktionen, mit denen der Min-Wert entsprechend dem Datentyp Elementweise ermittelt wird. Diese Funktion ist für Container mit Grundelementen verfügbar.

Z. B. bei einem TcVnPoint2_DINT werden der minimale X-Wert und der minimale Y-Wert unabhängig voneinander ermittelt, daher müssen die Werte nicht von dem gleichen Punkt stammen.

6.1.4.8.3.1 F_VN_MinElementElementwise_DINT



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nMin        : DINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nMin	DINT	Returns the requested element

Return value

HRESULT [▶ 150]

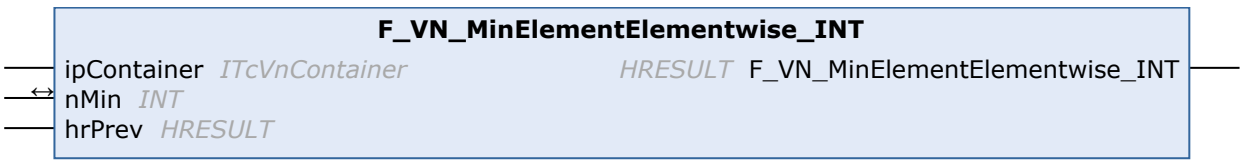
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.2 F_VN_MinElementElementwise_INT



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nMin        : INT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nMin	INT	Returns the requested element

Return value

HRESULT [▶ 150]

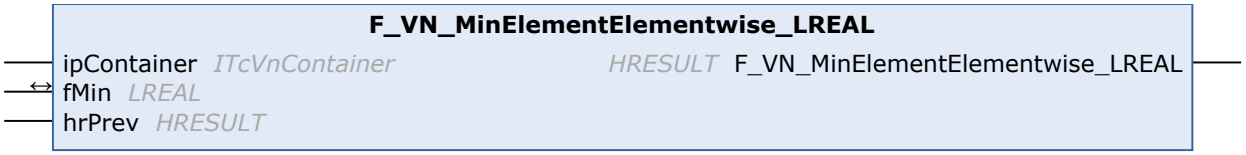
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.3 F_VN_MinElementElementwise_LREAL



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    fMin        : LREAL;
END_VAR
```

```
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fMin	LREAL	Returns the requested element

Return value

HRESULT [▶ 150]

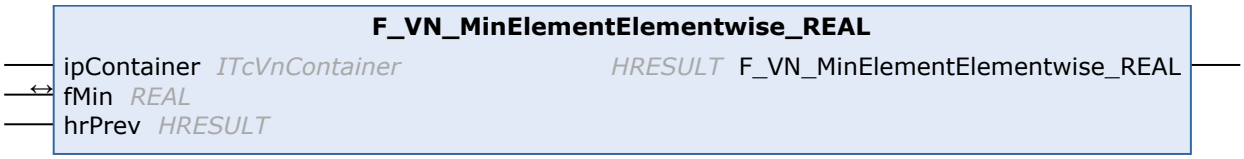
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.4 F_VN_MinElementElementwise_REAL



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    fMin        : REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fMin	REAL	Returns the requested element

Return value

HRESULT [▶ 150]

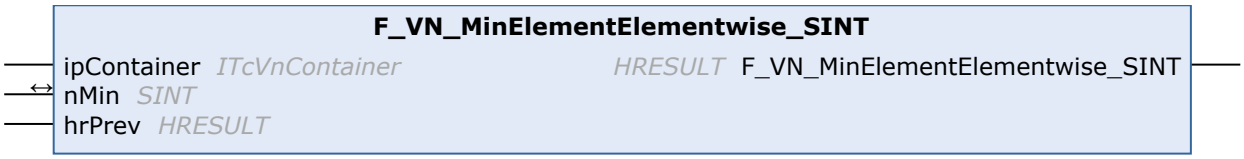
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.5 F_VN_MinElementElementwise_SINT



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nMin        : SINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nMin	SINT	Returns the requested element

Return value

HRESULT [▶ 150]

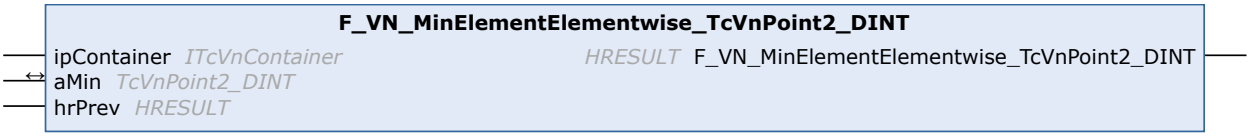
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.6 F_VN_MinElementElementwise_TcVnPoint2_DINT



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnPoint2_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnPoint2_DINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnPoint2_DINT [▶ 171]	Returns the requested element

Return value

HRESULT [▶ 150]

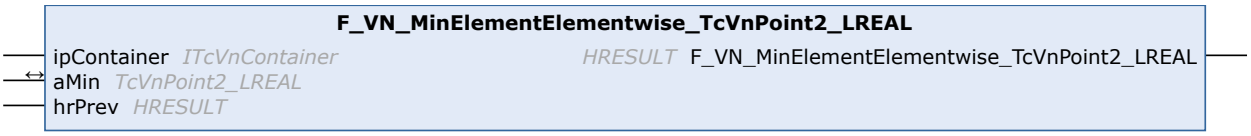
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.7 F_VN_MinElementElementwise_TcVnPoint2_LREAL



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnPoint2_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnPoint2_LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnPoint2_LREAL [▶ 171]	Returns the requested element

Return value

HRESULT [▶ 150]

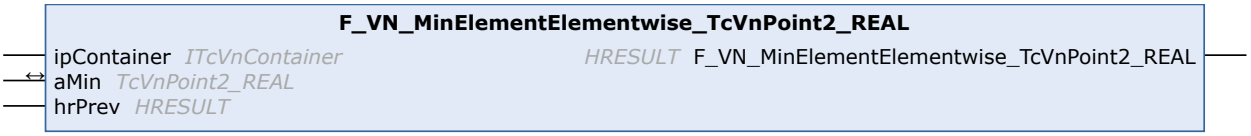
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.8 F_VN_MinElementElementwise_TcVnPoint2_REAL



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnPoint2_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnPoint2_REAL [▶ 171]	Returns the requested element

Return value

HRESULT [▶ 150]

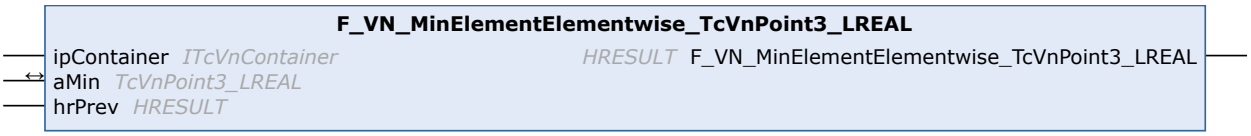
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.9 F_VN_MinElementElementwise_TcVnPoint3_LREAL



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnPoint3_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnPoint3_LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnPoint3_LREAL [▶ 171]	Returns the requested element

Return value

HRESULT [▶ 150]

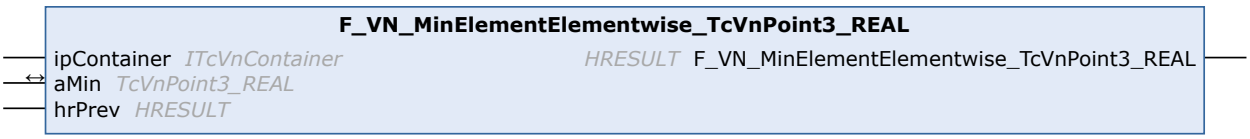
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.10 F_VN_MinElementElementwise_TcVnPoint3_REAL



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnPoint3_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnPoint3_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnPoint3_REAL [▶ 171]	Returns the requested element

Return value

HRESULT [▶ 150]

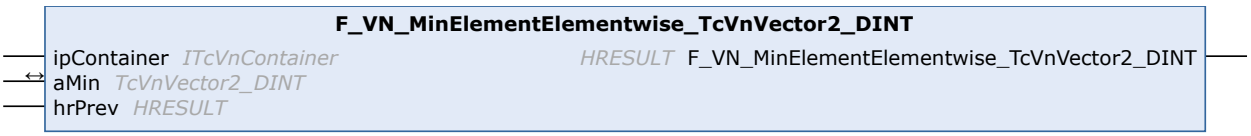
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.11 F_VN_MinElementElementwise_TcVnVector2_DINT



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnVector2_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnVector2_DINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector2_DINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

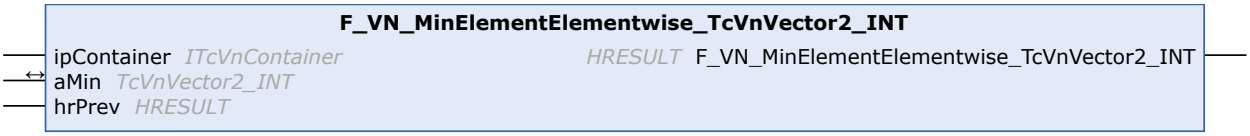
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.12 F_VN_MinElementElementwise_TcVnVector2_INT



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnVector2_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnVector2_INT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector2_INT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

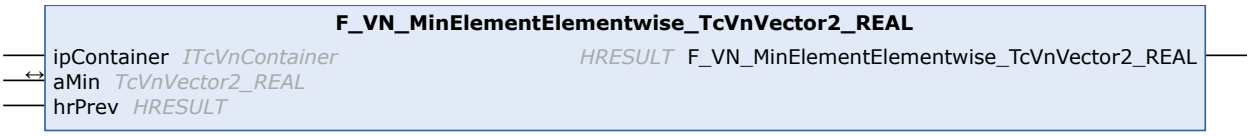
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.13 F_VN_MinElementElementwise_TcVnVector2_REAL



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnVector2_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnVector2_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector2_REAL [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

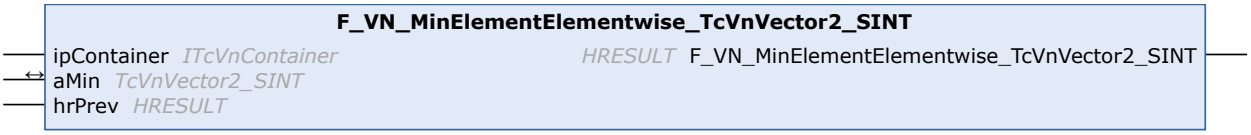
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.14 F_VN_MinElementElementwise_TcVnVector2_SINT



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnVector2_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnVector2_SINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector2_SINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

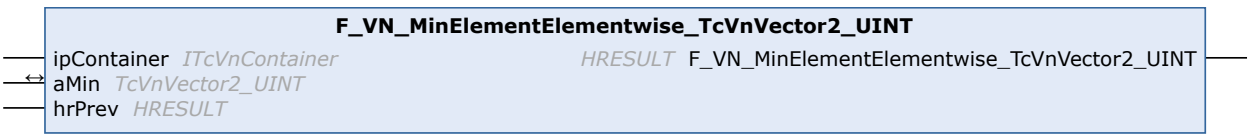
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.15 F_VN_MinElementElementwise_TcVnVector2_UINT



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnVector2_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnVector2_UINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector2_UINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

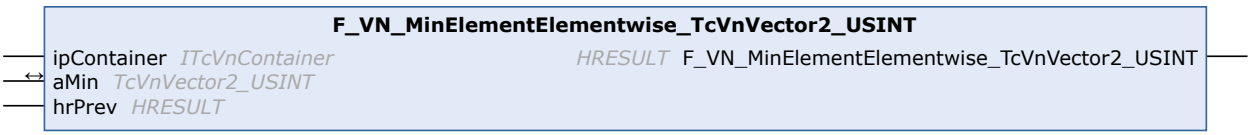
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.16 F_VN_MinElementElementwise_TcVnVector2_USINT



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnVector2_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnVector2_USINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector2_USINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

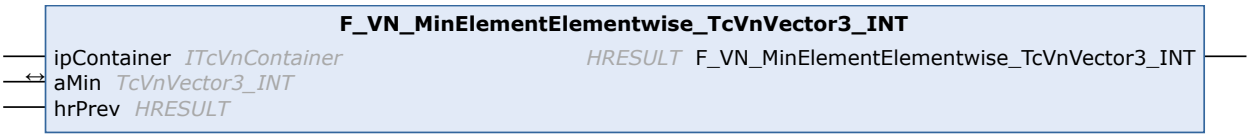
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.17 F_VN_MinElementElementwise_TcVnVector3_INT



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnVector3_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnVector3_INT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector3_INT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

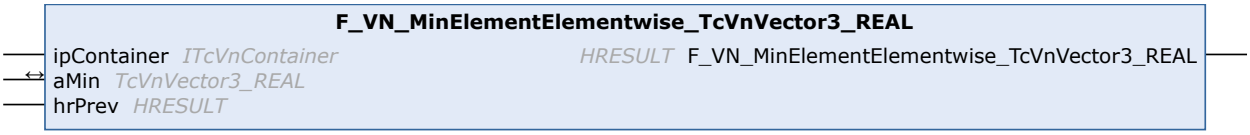
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.18 F_VN_MinElementElementwise_TcVnVector3_REAL



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnVector3_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnVector3_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector3_REAL [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

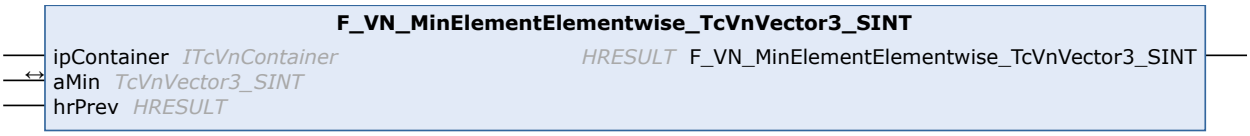
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.19 F_VN_MinElementElementwise_TcVnVector3_SINT



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnVector3_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnVector3_SINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector3_SINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

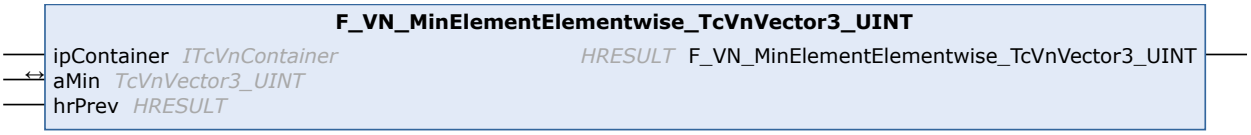
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.20 F_VN_MinElementElementwise_TcVnVector3_UINT



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnVector3_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnVector3_UINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector3_UINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

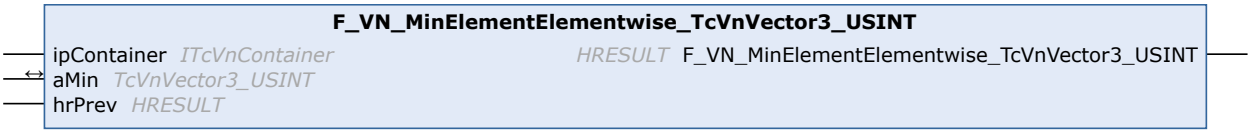
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.21 F_VN_MinElementElementwise_TcVnVector3_USINT



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnVector3_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnVector3_USINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector3_USINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

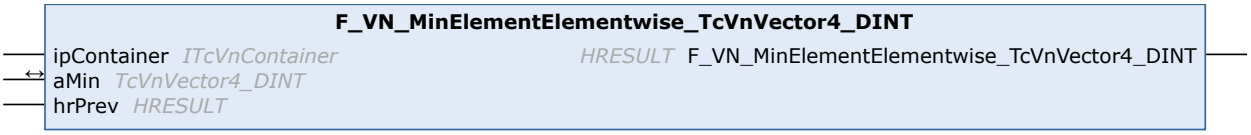
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.22 F_VN_MinElementElementwise_TcVnVector4_DINT



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnVector4_DINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnVector4_DINT;
END_VAR
```

```
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector4_DINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

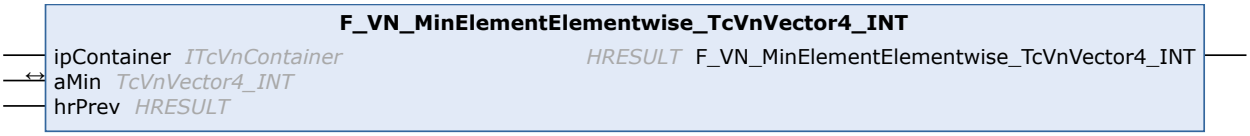
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.23 F_VN_MinElementElementwise_TcVnVector4_INT



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnVector4_INT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnVector4_INT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector4_INT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

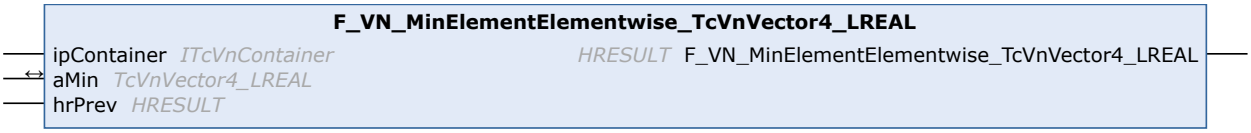
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.24 F_VN_MinElementElementwise_TcVnVector4_LREAL



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnVector4_LREAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector4_LREAL [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

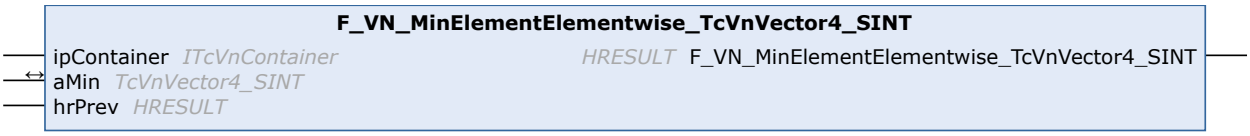
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.25 F_VN_MinElementElementwise_TcVnVector4_SINT



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnVector4_SINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnVector4_SINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector4_SINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

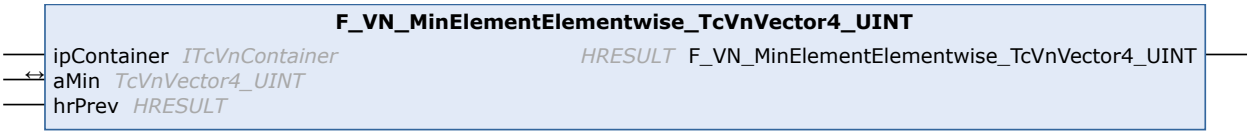
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.26 F_VN_MinElementElementwise_TcVnVector4_UINT



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnVector4_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnVector4_UINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector4_UINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

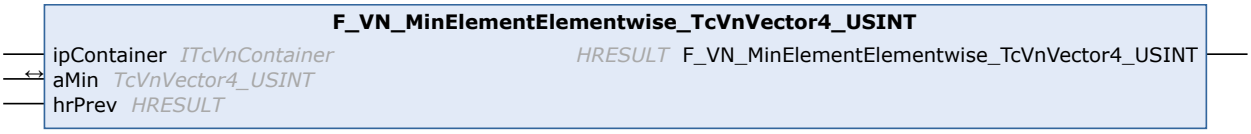
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.27 F_VN_MinElementElementwise_TcVnVector4_USINT



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnVector4_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnVector4_USINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector4_USINT [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

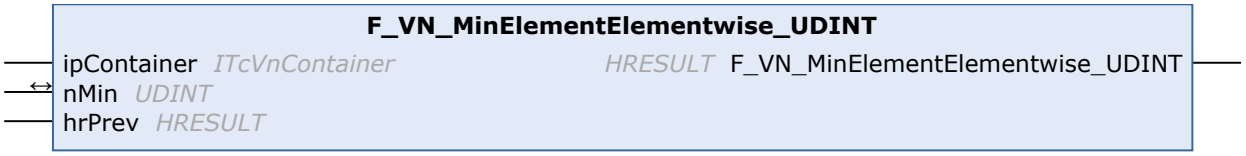
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.28 F_VN_MinElementElementwise_UDINT



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_UDINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nMin        : UDINT;
END_VAR
```

```
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nMin	UDINT	Returns the requested element

Return value

HRESULT [▶ 150]

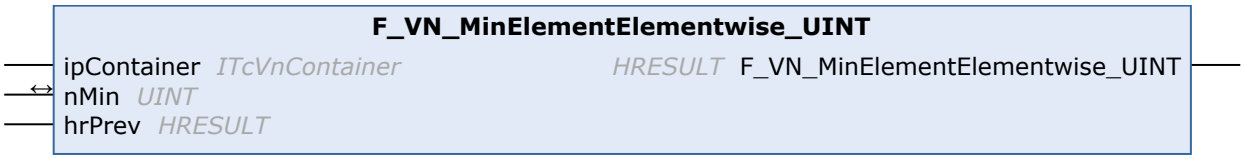
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.29 F_VN_MinElementElementwise_UINT



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_UINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nMin        : UINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nMin	UINT	Returns the requested element

Return value

HRESULT [▶ 150]

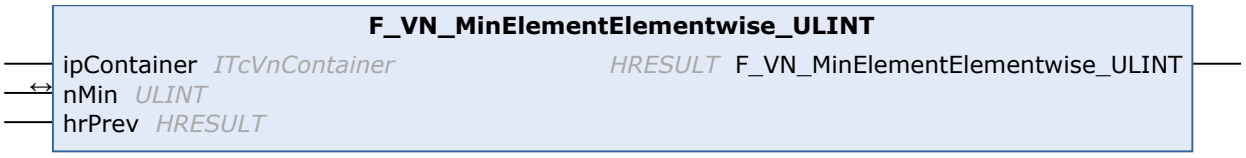
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.30 F_VN_MinElementElementwise_ULINT



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_ULINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nMin        : ULINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nMin	ULINT	Returns the requested element

Return value

HRESULT [▶ 150]

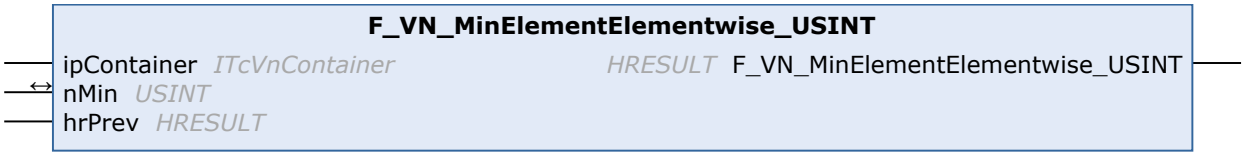
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.31 F_VN_MinElementElementwise_USINT



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_USINT : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    nMin        : USINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nMin	USINT	Returns the requested element

Return value

HRESULT [▶ 150]

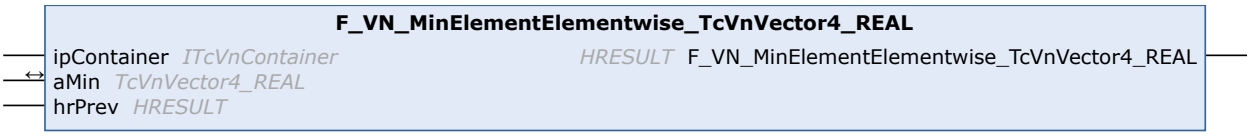
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.3.32 F_VN_MinElementElementwise_TcVnVector4_REAL



Gets the element wise minimum container element.

Syntax

Definition:

```
FUNCTION F_VN_MinElementElementwise_TcVnVector4_REAL : HRESULT
VAR_INPUT
    ipContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aMin        : TcVnVector4_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer	ITcVnContainer [▶ 385]	Source container
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMin	TcVnVector4_REAL [▶ 173]	Returns the requested element

Return value

HRESULT [▶ 150]

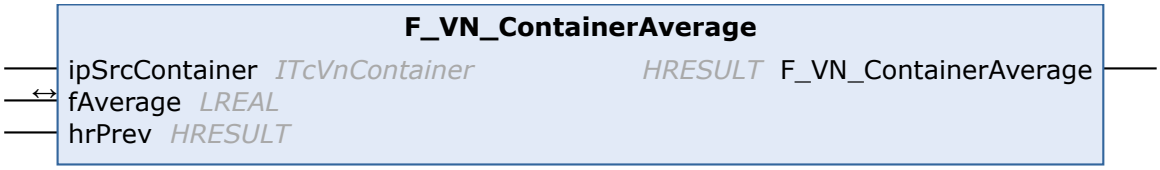
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.4 F_VN_ContainerAverage



Calculate the average of elements in a container.

Syntax

Definition:

```
FUNCTION F_VN_ContainerAverage : HRESULT
VAR_INPUT
    ipSrcContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    fAverage       : LREAL;
END_VAR
VAR_INPUT
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer	ITcVnContainer [▶ 385]	Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fAverage	LREAL	Returns the calculated average value

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Folgende Tabelle zeigt, welche Funktionen für welchen Container-Typ verwendet werden können:

Typen von Containern	Benutzbare Funktionen
Container von Elementen	F_VN_ContainerAverage [▶ 973] F_VN_ContainerAverageVariance [▶ 977]
Container von Vektoren mit je 2 Elementen	F_VN_ContainerAverageElementwise2 [▶ 974] F_VN_ContainerAverageVarianceElementwise2 [▶ 978]
Container von Vektoren mit je 3 Elementen	F_VN_ContainerAverageElementwise3 [▶ 975] F_VN_ContainerAverageVarianceElementwise3 [▶ 980]
Container von Vektoren mit je 4 Elementen	F_VN_ContainerAverageElementwise4 [▶ 976] F_VN_ContainerAverageVarianceElementwise4 [▶ 981]

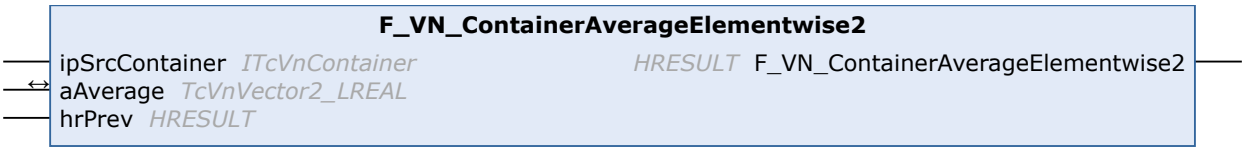
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.5 F_VN_ContainerAverageElementwise2



Calculate the elementwise average of 2D elements in a container.

Syntax

Definition:

```
FUNCTION F_VN_ContainerAverageElementwise2 : HRESULT
VAR_INPUT
    ipSrcContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aAverage       : TcVnVector2_LREAL;
END_VAR
VAR_INPUT
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer	ITcVnContainer [▶ 385]	Source container with 2D points or vectors
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAverage	TcVnVector2_LREAL [▶ 173]	Returns the calculated average values

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Diese Funktion berechnet elementweise die Durchschnittswerte für einen Container mit zwei elementaren Sub-Elementen pro Container-Element.

Folgende Tabelle zeigt, welche Funktionen für welchen Container-Typ verwendet werden können:

Typen von Containern	Benutzbare Funktionen
Container von Elementen	F_VN_ContainerAverage [▶ 973] F_VN_ContainerAverageVariance [▶ 977]
Container von Vektoren mit je 2 Elementen	F_VN_ContainerAverageElementwise2 [▶ 974] F_VN_ContainerAverageVarianceElementwise2 [▶ 978]
Container von Vektoren mit je 3 Elementen	F_VN_ContainerAverageElementwise3 [▶ 975] F_VN_ContainerAverageVarianceElementwise3 [▶ 980]
Container von Vektoren mit je 4 Elementen	F_VN_ContainerAverageElementwise4 [▶ 976] F_VN_ContainerAverageVarianceElementwise4 [▶ 981]

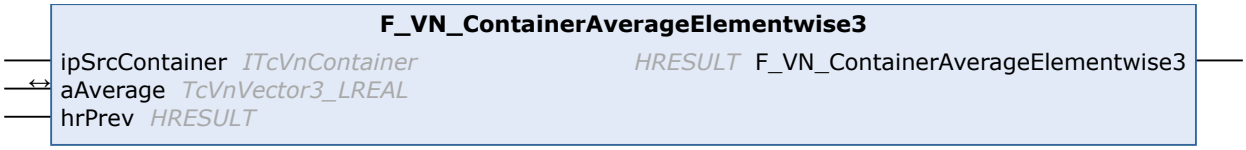
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.6 F_VN_ContainerAverageElementwise3



Calculate the elementwise average of 3D elements in a container.

Syntax

Definition:

```
FUNCTION F_VN_ContainerAverageElementwise3 : HRESULT
VAR_INPUT
    ipSrcContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aAverage       : TcVnVector3_LREAL;
END_VAR
VAR_INPUT
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer	ITcVnContainer [▶ 385]	Source container with 3D points or vectors
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAverage	TcVnVector3_LREAL [▶ 173]	Returns the calculated average values

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Folgende Tabelle zeigt, welche Funktionen für welchen Container-Typ verwendet werden können:

Typen von Containern	Benutzbare Funktionen
Container von Elementen	F_VN_ContainerAverage [▶ 973] F_VN_ContainerAverageVariance [▶ 977]
Container von Vektoren mit je 2 Elementen	F_VN_ContainerAverageElementwise2 [▶ 974] F_VN_ContainerAverageVarianceElementwise2 [▶ 978]
Container von Vektoren mit je 3 Elementen	F_VN_ContainerAverageElementwise3 [▶ 975] F_VN_ContainerAverageVarianceElementwise3 [▶ 980]
Container von Vektoren mit je 4 Elementen	F_VN_ContainerAverageElementwise4 [▶ 976] F_VN_ContainerAverageVarianceElementwise4 [▶ 981]

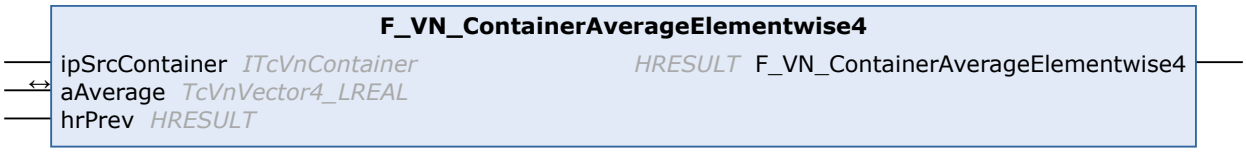
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.7 F_VN_ContainerAverageElementwise4



Calculate the elementwise average of 4D elements in a container.

Syntax

Definition:

```
FUNCTION F_VN_ContainerAverageElementwise4 : HRESULT
VAR_INPUT
    ipSrcContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aAverage       : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer	ITcVnContainer [▶ 385]	Source container with 4D vectors
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAverage	TcVnVector4_LREAL [▶ 173]	Returns the calculated average values

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Folgende Tabelle zeigt, welche Funktionen für welchen Container-Typ verwendet werden können:

Typen von Containern	Benutzbare Funktionen
Container von Elementen	F_VN_ContainerAverage [▶ 973] F_VN_ContainerAverageVariance [▶ 977]
Container von Vektoren mit je 2 Elementen	F_VN_ContainerAverageElementwise2 [▶ 974] F_VN_ContainerAverageVarianceElementwise2 [▶ 978]
Container von Vektoren mit je 3 Elementen	F_VN_ContainerAverageElementwise3 [▶ 975] F_VN_ContainerAverageVarianceElementwise3 [▶ 980]
Container von Vektoren mit je 4 Elementen	F_VN_ContainerAverageElementwise4 [▶ 976] F_VN_ContainerAverageVarianceElementwise4 [▶ 981]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.8 F_VN_ContainerAverageVariance

Calculate the average and variance of elements in a container.

Syntax

Definition:

```
FUNCTION F_VN_ContainerAverageVariance : HRESULT
VAR_INPUT
    ipSrcContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    fAverage       : LREAL;
    fVariance      : LREAL;
END_VAR
```

```
VAR_INPUT
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer	ITcVnContainer [▶ 385]	Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fAverage	LREAL	Returns the calculated average value
fVariance	LREAL	Returns the calculated variance value

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Folgende Tabelle zeigt, welche Funktionen für welchen Container-Typ verwendet werden können:

Typen von Containern	Benutzbare Funktionen
Container von Elementen	F_VN_ContainerAverage [▶ 973] F_VN_ContainerAverageVariance [▶ 977]
Container von Vektoren mit je 2 Elementen	F_VN_ContainerAverageElementwise2 [▶ 974] F_VN_ContainerAverageVarianceElementwise2 [▶ 978]
Container von Vektoren mit je 3 Elementen	F_VN_ContainerAverageElementwise3 [▶ 975] F_VN_ContainerAverageVarianceElementwise3 [▶ 980]
Container von Vektoren mit je 4 Elementen	F_VN_ContainerAverageElementwise4 [▶ 976] F_VN_ContainerAverageVarianceElementwise4 [▶ 981]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.9 F_VN_ContainerAverageVarianceElementwise2

Calculate the elementwise average and variance of 2D elements in a container.

Syntax

Definition:

```
FUNCTION F_VN_ContainerAverageVarianceElementwise2 : HRESULT
VAR_INPUT
    ipSrcContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aAverage       : TcVnVector2_LREAL;
    aVariance      : TcVnVector2_LREAL;
END_VAR
VAR_INPUT
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer	ITcVnContainer [▶ 385]	Source container with 2D points or vectors
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAverage	TcVnVector2_LREAL [▶ 173]	Returns the calculated average values
aVariance	TcVnVector2_LREAL [▶ 173]	Returns the calculated variance values

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Folgende Tabelle zeigt, welche Funktionen für welchen Container-Typ verwendet werden können:

Typen von Containern	Benutzbare Funktionen
Container von Elementen	F_VN_ContainerAverage [▶ 973] F_VN_ContainerAverageVariance [▶ 977]
Container von Vektoren mit je 2 Elementen	F_VN_ContainerAverageElementwise2 [▶ 974] F_VN_ContainerAverageVarianceElementwise2 [▶ 978]
Container von Vektoren mit je 3 Elementen	F_VN_ContainerAverageElementwise3 [▶ 975] F_VN_ContainerAverageVarianceElementwise3 [▶ 980]
Container von Vektoren mit je 4 Elementen	F_VN_ContainerAverageElementwise4 [▶ 976] F_VN_ContainerAverageVarianceElementwise4 [▶ 981]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.10 F_VN_ContainerAverageVarianceElementwise3

Calculate the elementwise average and variance of 3D elements in a container.

Syntax

Definition:

```
FUNCTION F_VN_ContainerAverageVarianceElementwise3 : HRESULT
VAR_INPUT
    ipSrcContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aAverage       : TcVnVector3_LREAL;
    aVariance      : TcVnVector3_LREAL;
END_VAR
VAR_INPUT
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer	ITcVnContainer [▶ 385]	Source container with 3D points or vectors
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAverage	TcVnVector3_LREAL [▶ 173]	Returns the calculated average values
aVariance	TcVnVector3_LREAL [▶ 173]	Returns the calculated variance values

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Folgende Tabelle zeigt, welche Funktionen für welchen Container-Typ verwendet werden können:

Typen von Containern	Benutzbare Funktionen
Container von Elementen	F_VN_ContainerAverage [▶ 973] F_VN_ContainerAverageVariance [▶ 977]
Container von Vektoren mit je 2 Elementen	F_VN_ContainerAverageElementwise2 [▶ 974] F_VN_ContainerAverageVarianceElementwise2 [▶ 978]
Container von Vektoren mit je 3 Elementen	F_VN_ContainerAverageElementwise3 [▶ 975] F_VN_ContainerAverageVarianceElementwise3 [▶ 980]
Container von Vektoren mit je 4 Elementen	F_VN_ContainerAverageElementwise4 [▶ 976] F_VN_ContainerAverageVarianceElementwise4 [▶ 981]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.11 F_VN_ContainerAverageVarianceElementwise4

Calculate the elementwise average and variance of 4D elements in a container.

Syntax

Definition:

```
FUNCTION F_VN_ContainerAverageVarianceElementwise4 : HRESULT
VAR_INPUT
    ipSrcContainer : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aAverage       : TcVnVector4_LREAL;
    aVariance      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer	ITcVnContainer [▶ 385]	Source container with 4D vectors
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAverage	TcVnVector4_LREAL [▶ 173]	Returns the calculated average values
aVariance	TcVnVector4_LREAL [▶ 173]	Returns the calculated variance values

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Folgende Tabelle zeigt, welche Funktionen für welchen Container-Typ verwendet werden können:

Typen von Containern	Benutzbare Funktionen
Container von Elementen	F_VN_ContainerAverage [▶ 973] F_VN_ContainerAverageVariance [▶ 977]
Container von Vektoren mit je 2 Elementen	F_VN_ContainerAverageElementwise2 [▶ 974] F_VN_ContainerAverageVarianceElementwise2 [▶ 978]
Container von Vektoren mit je 3 Elementen	F_VN_ContainerAverageElementwise3 [▶ 975] F_VN_ContainerAverageVarianceElementwise3 [▶ 980]
Container von Vektoren mit je 4 Elementen	F_VN_ContainerAverageElementwise4 [▶ 976] F_VN_ContainerAverageVarianceElementwise4 [▶ 981]

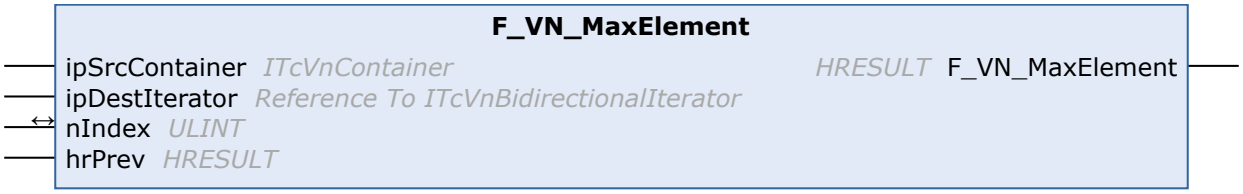
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.12 F_VN_MaxElement



Gets the maximum element of a container.

Syntax

Definition:

```
FUNCTION F_VN_MaxElement : HRESULT
VAR_INPUT
    ipSrcContainer : ITcVnContainer;
    ipDestIterator : Reference To ITcVnBidirectionalIterator;
END_VAR
VAR_IN_OUT
    nIndex         : ULINT;
```

```
END_VAR
VAR_INPUT
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer	ITcVnContainer [▶ 385]	Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)
ipDestIterator	Reference To ITcVnBidirectionalIterator [▶ 375]	Returns an iterator to the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nIndex	ULINT	Returns the position of the requested element within the container

Return value

HRESULT [▶ 150]

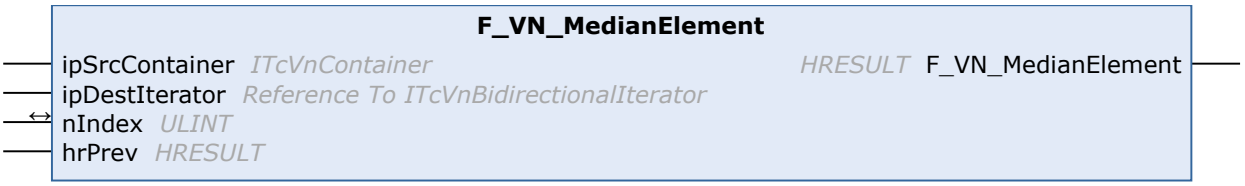
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.13 F_VN_MedianElement



Gets the median element of a container.

Syntax

Definition:

```
FUNCTION F_VN_MedianElement : HRESULT
VAR_INPUT
    ipSrcContainer : ITcVnContainer;
    ipDestIterator : Reference To ITcVnBidirectionalIterator;
END_VAR
VAR_IN_OUT
    nIndex          : ULINT;
END_VAR
VAR_INPUT
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer	ITcVnContainer [▶ 385]	Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)
ipDestIterator	Reference To ITcVnBidirectionalIterator [▶ 375]	Returns an iterator to the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nIndex	ULINT	Returns the position of the requested element within the container

Return value

HRESULT [▶ 150]

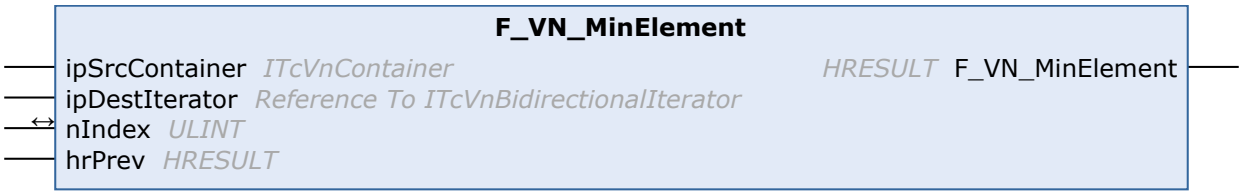
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.8.14 F_VN_MinElement



Gets the minimum element of a container.

Syntax

Definition:

```
FUNCTION F_VN_MinElement : HRESULT
VAR_INPUT
    ipSrcContainer : ITcVnContainer;
    ipDestIterator : Reference To ITcVnBidirectionalIterator;
END_VAR
VAR_IN_OUT
    nIndex         : ULINT;
END_VAR
VAR_INPUT
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContainer	ITcVnContainer [▶ 385]	Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)
ipDestIterator	Reference To ITcVnBidirectionalIterator [▶ 375]	Returns an iterator to the requested element
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nIndex	ULINT	Returns the position of the requested element within the container

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9 Contour Analysis

Die Gruppe enthält Funktionen zur Analyse von Konturen [▶ 165]:

Funktionen

Formen approximieren

- F_VN_ApproximatePolygon [▶ 986]
- F_VN_ConvexHullPoints(Exp) [▶ 1001]
- F_VN_EnclosingCircle [▶ 1006]
- F_VN_EnclosingRectangle [▶ 1007]
- F_VN_EnclosingTriangle [▶ 1008]
- F_VN_FitEllipse [▶ 1009]
- F_VN_FitLine(Exp) [▶ 1011]
- F_VN_UprightBoundingRectangle [▶ 1018]

Punkte

- F_VN_CheckIfPointIsInsideContour [▶ 987]
- F_VN_ContourExtremePoint [▶ 994]
- F_VN_ConvexityDefects [▶ 1003]

Geometrische Merkmale

- F_VN_ContourArea [▶ 988]
- F_VN_ContourCenterOfMass [▶ 989]
- F_VN_ContourCircularity [▶ 990]
- F_VN_ContourConvexity [▶ 991]
- F_VN_ContourEccentricity [▶ 992]
- F_VN_ContourElongation [▶ 993]
- F_VN_ContourInertiaRatio [▶ 995]
- F_VN_ContourMoments [▶ 996]
- F_VN_ContourOrientation(Exp) [▶ 997]
- F_VN_ContourPerimeter [▶ 999]
- F_VN_ContourRoundness [▶ 1000]
- F_VN_FourierDescriptors [▶ 1012]

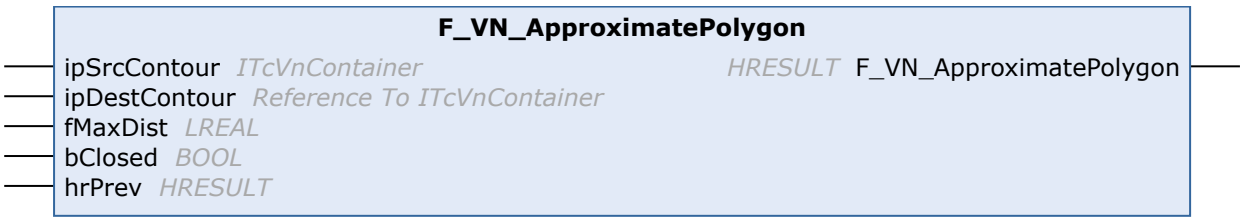
Matching

- F_VN_MatchContours(Exp) [▶ 1017]
- F_VN_MatchContours1vsN(Exp) [▶ 1016]

Benutzerdefinierte Operationen

- F_VN_CustomElementWiseContainerOperation_ITcVnContainer [▶ 1004]
- F_VN_CustomElementWiseContainerOperation_ITcVnForwardIterator [▶ 1005]

6.1.4.9.1 F_VN_ApproximatePolygon



Approximate a contour to a simplified polygon (using the Douglas-Peucker algorithm).

Syntax

Definition:

```
FUNCTION F_VN_ApproximatePolygon : HRESULT
VAR_INPUT
    ipSrcContour  : ITcVnContainer;
    ipDestContour : Reference To ITcVnContainer;
    fMaxDist      : LREAL;
    bClosed       : BOOL;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcContour	ITcVnContainer [▶ 385]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
ipDestContour	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the approximated polygon points (same type ID as ipSrcContour; Non-zero interface pointers are reused.)
fMaxDist	LREAL	Maximum distance between the original contour and its approximation
bClosed	BOOL	Specify, if the contour is closed (first and last points connected) or not
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.2 F_VN_CheckIfPointIsInsideContour

Checks if a point is inside a contour (and optionally return the distance to it).

Syntax

Definition:

```
FUNCTION F_VN_CheckIfPointIsInsideContour : HRESULT
VAR_INPUT
    ipContour        : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aPoint           : TcVnPoint2_REAL;
    fDist            : LREAL;
END_VAR
VAR_INPUT
    bMeasureDistance : BOOL;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContour	ITcVnContainer [▶ 385]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL)
bMeasureDistance	BOOL	If true, fDist returns the distance to the nearest contour edge. Otherwise, fDist only returns -1, 0 or +1 (recommended to set to false if the distance is not required, as this is faster).
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aPoint	TcVnPoint2_REAL [▶ 171]	Point position to check
fDist	LREAL	Returns the result (> 0: the point is inside the contour; 0: the point is on the contour; < 0: the point is outside the contour)

Return value

HRESULT [▶ 150]

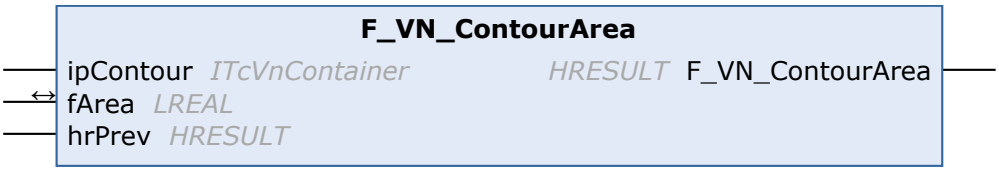
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.3 F_VN_ContourArea



Estimate the contour area using Green's formula.

Syntax

Definition:

```
FUNCTION F_VN_ContourArea : HRESULT
VAR_INPUT
    ipContour : ITcVnContainer;
END_VAR
VAR_IN_OUT
    fArea     : LREAL;
END_VAR
VAR_INPUT
    hrPrev    : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContour	ITcVnContainer [▶ 385]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fArea	LREAL	Returns the estimated contour area (The actual contour area may differ depending on the shape of the contour.)

Return value

HRESULT [▶ 150]

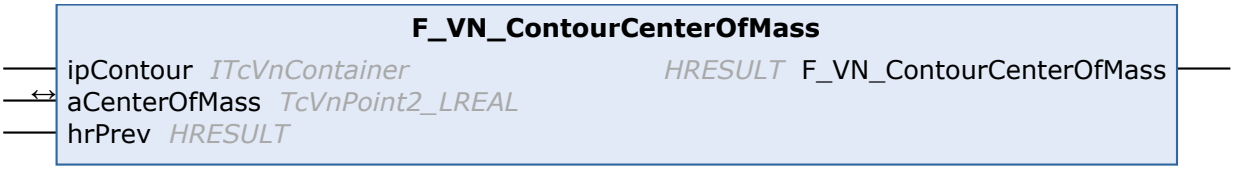
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.4 F_VN_ContourCenterOfMass



Computes the center of mass of a contour.

Syntax

Definition:

```
FUNCTION F_VN_ContourCenterOfMass : HRESULT
VAR_INPUT
    ipContour     : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aCenterOfMass : TcVnPoint2_LREAL;
END_VAR
VAR_INPUT
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContour	ITcVnContainer [▶ 385]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCenterOfMass	TcVnPoint2_LREAL [▶ 171]	Returns the center of mass of the contour

Return value

HRESULT [▶ 150]

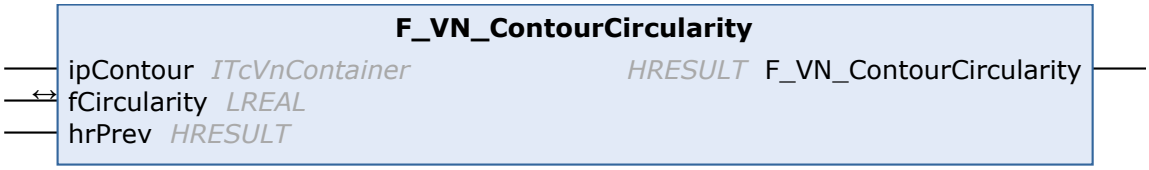
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.5 F_VN_ContourCircularity



Computes the circularity of a contour (1.0: ideal circle, 0.0: straight line).

Syntax

Definition:

```
FUNCTION F_VN_ContourCircularity : HRESULT
VAR_INPUT
    ipContour    : ITcVnContainer;
END_VAR
VAR_IN_OUT
    fCircularity : LREAL;
END_VAR
VAR_INPUT
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContour	ITcVnContainer [▶ 385]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fCircularity	LREAL	Returns the circularity of the contour [0..1]

Return value

HRESULT [▶ 150]

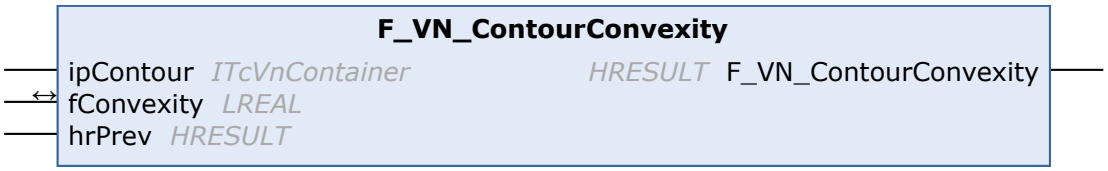
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.6 F_VN_ContourConvexity



Computes the convexity of a contour as the ratio between the contour's area and the area of the convex hull (1.0: convex shapes (i.e. rectangle, circle, ..., etc.), 0.0: zero area (i.e. a line or an empty contour)).

Syntax

Definition:

```
FUNCTION F_VN_ContourConvexity : HRESULT
VAR_INPUT
    ipContour  : ITcVnContainer;
END_VAR
VAR_IN_OUT
    fConvexity : LREAL;
END_VAR
VAR_INPUT
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContour	ITcVnContainer [▶ 385]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fConvexity	LREAL	Returns the convexity of the contour [0..1]

Return value

HRESULT [▶ 150]

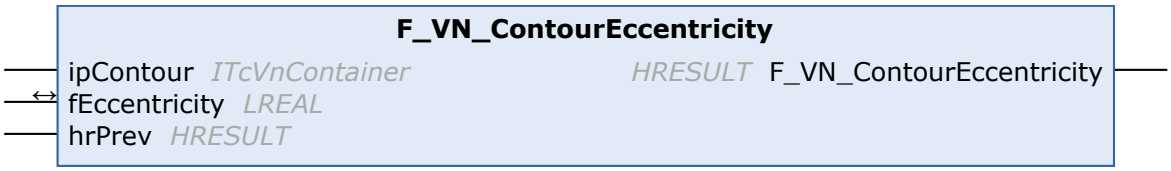
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.7 F_VN_ContourEccentricity



Computes the eccentricity of a contour (0.0: circular, 1.0: linear).

Syntax

Definition:

```
FUNCTION F_VN_ContourEccentricity : HRESULT
VAR_INPUT
    ipContour     : ITcVnContainer;
END_VAR
VAR_IN_OUT
    fEccentricity : LREAL;
END_VAR
VAR_INPUT
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContour	ITcVnContainer [▶ 385]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fEccentricity	LREAL	Returns the eccentricity of the contour [0..1]

Return value

HRESULT [▶ 150]

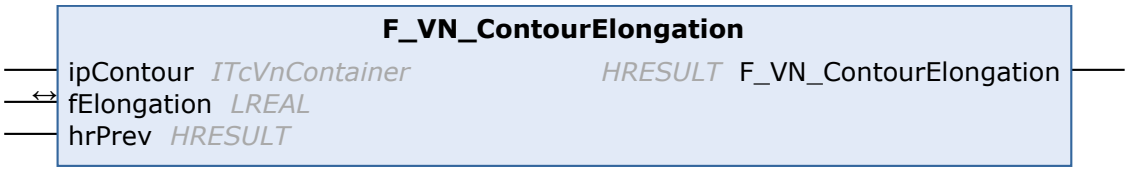
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.8 F_VN_ContourElongation



Computes the elongation factor of a contour.

Syntax

Definition:

```
FUNCTION F_VN_ContourElongation : HRESULT
VAR_INPUT
    ipContour   : ITcVnContainer;
END_VAR
VAR_IN_OUT
    fElongation : LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContour	ITcVnContainer [▶ 385]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fElongation	LREAL	Returns the elongation of the contour

Return value

HRESULT [▶ 150]

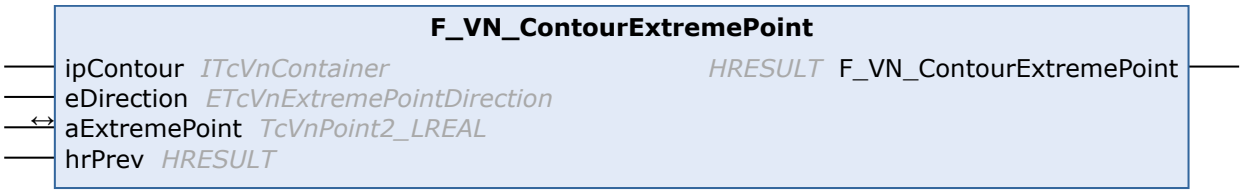
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.9 F_VN_ContourExtremePoint



Finds the contour extreme point.

Syntax

Definition:

```
FUNCTION F_VN_ContourExtremePoint : HRESULT
VAR_INPUT
    ipContour     : ITcVnContainer;
    eDirection    : ETcVnExtremePointDirection;
END_VAR
VAR_IN_OUT
    aExtremePoint : TcVnPoint2_LREAL;
END_VAR
VAR_INPUT
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContour	ITcVnContainer [▶ 385]	Source container with 2D points (TcVnPoint2_DINT or TcVnPoint2_REAL or TcVnPoint2_LREAL)
eDirection	ETcVnExtremePointDirection [▶ 211]	Selects the search direction for the extreme point
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aExtremePoint	TcVnPoint2_LREAL [▶ 171]	Returns the extreme point of the contour, according to eDirection

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.10 F_VN_ContourInertiaRatio

Computes the inertia ratio of a contour which reflects how a contour/shape is elongated (i.e. circle: 1.0, line: 0.0).

Syntax

Definition:

```
FUNCTION F_VN_ContourInertiaRatio : HRESULT
VAR_INPUT
    ipContour     : ITcVnContainer;
END_VAR
VAR_IN_OUT
    fInertiaRatio : LREAL;
END_VAR
VAR_INPUT
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContour	ITcVnContainer [▶ 385]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fInertiaRatio	LREAL	Returns the inertia raio of the contour [0..1]

Return value

HRESULT [▶ 150]

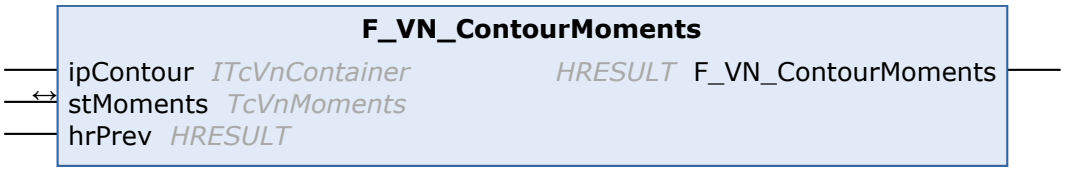
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.11 F_VN_ContourMoments



Computes the spatial moments, the central moments, and the central normalized moments of a contour up to the third order.

Syntax

Definition:

```
FUNCTION F_VN_ContourMoments : HRESULT
VAR_INPUT
    ipContour : ITcVnContainer;
END_VAR
VAR_IN_OUT
    stMoments : TcVnMoments;
END_VAR
VAR_INPUT
    hrPrev    : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContour	ITcVnContainer [▶ 385]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stMoments	TcVnMoments [▶ 245]	Returns a struct containing the moments

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.12 F_VN_ContourOrientation

Calculate the orientation of a contour or a set of points.

Syntax

Definition:

```
FUNCTION F_VN_ContourOrientation : HRESULT
VAR_INPUT
    ipContour     : ITcVnContainer;
END_VAR
VAR_IN_OUT
    stOrientation : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContour	ITcVnContainer [▶ 385]	Source point set (ContainerType_Vector_TcVnPoint2_DINT, ContainerType_Vector_TcVnPoint2_REAL, or ContainerType_Vector_TcVnPoint2_LREAL. The elements of this container are the individual points.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stOrientation	TcVnRotatedRectangle [▶ 258]	Resulting rotated rectangle, containig center of mass, lenghts of axes, and rotation angle of the contour in clockwise direction.

Return value

HRESULT [▶ 150]

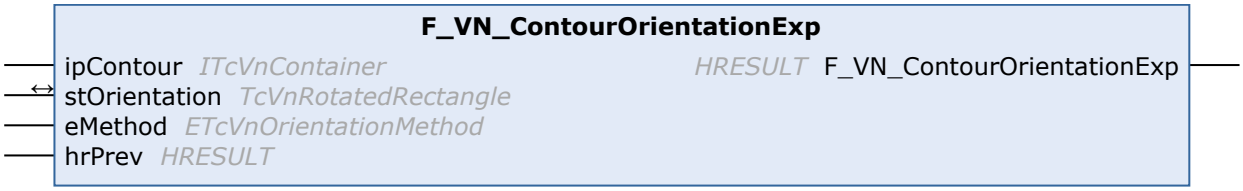
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.13 F_VN_ContourOrientationExp



Calculate the orientation of a contour or a set of points. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_ContourOrientationExp : HRESULT
VAR_INPUT
    ipContour     : ITcVnContainer;
END_VAR
VAR_IN_OUT
    stOrientation : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    eMethod       : ETcVnOrientationMethod;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContour	ITcVnContainer [▶ 385]	Source point set (ContainerType_Vector_TcVnPoint2_DINT, ContainerType_Vector_TcVnPoint2_REAL, or ContainerType_Vector_TcVnPoint2_LREAL. The elements of this container are the individual points.)
eMethod	ETcVnOrientationMethod [▶ 223]	Method for calculating the orientation.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stOrientation	TcVnRotatedRectangle [▶ 258]	Resulting rotated rectangle, containig center of mass, lenghts of axes, and rotation angle of the contour in clockwise direction.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.14 F_VN_ContourPerimeter

Computes the perimeter of a contour (curve length if the contour is not closed).

Syntax

Definition:

```
FUNCTION F_VN_ContourPerimeter : HRESULT
VAR_INPUT
    ipContour  : ITcVnContainer;
END_VAR
VAR_IN_OUT
    fPerimeter : LREAL;
END_VAR
VAR_INPUT
    bClosed    : BOOL;
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContour	ITcVnContainer [▶ 385]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
bClosed	BOOL	Specifys, if the contour is closed (first and last points connected) or not
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fPerimeter	LREAL	Returns the perimeter of the contour

Return value

HRESULT [▶ 150]

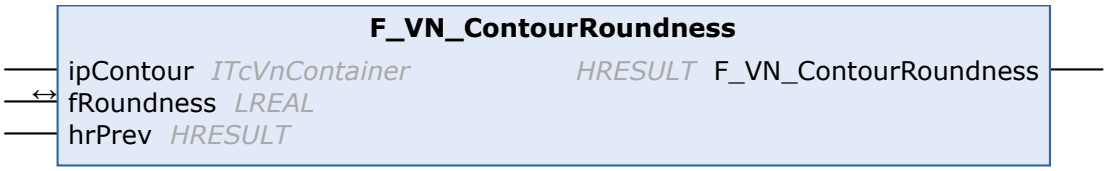
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.15 F_VN_ContourRoundness



Computes the roundness of a contour (perimeter^2 / area).

Syntax

Definition:

```
FUNCTION F_VN_ContourRoundness : HRESULT
VAR_INPUT
    ipContour  : ITcVnContainer;
END_VAR
VAR_IN_OUT
    fRoundness : LREAL;
END_VAR
VAR_INPUT
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContour	ITcVnContainer [▶ 385]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fRoundness	LREAL	Returns the roundness of the contour

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.16 F_VN_ConvexHullPoints

Determines the convex hull of a point set.

Syntax

Definition:

```
FUNCTION F_VN_ConvexHullPoints : HRESULT
VAR_INPUT
    ipPointSet   : ITcVnContainer;
    ipConvexHull : Reference To ITcVnContainer;
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipPointSet	ITcVnContainer [▶ 385]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
ipConvexHull	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the sorted convex hull points (same type ID as ipPointSet; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.17 F_VN_ConvexHullPointsExp

Determines the convex hull of a point set. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_ConvexHullPointsExp : HRESULT
VAR_INPUT
    ipPointSet   : ITcVnContainer;
    ipConvexHull : Reference To ITcVnContainer;
    bClockwise   : BOOL;
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipPointSet	ITcVnContainer [▶ 385]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
ipConvexHull	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the sorted convex hull points (same type ID as ipPointSet; Non-zero interface pointers are reused.)
bClockwise	BOOL	Selects the sorting direction of the hull points, assuming cartesian coordinates.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

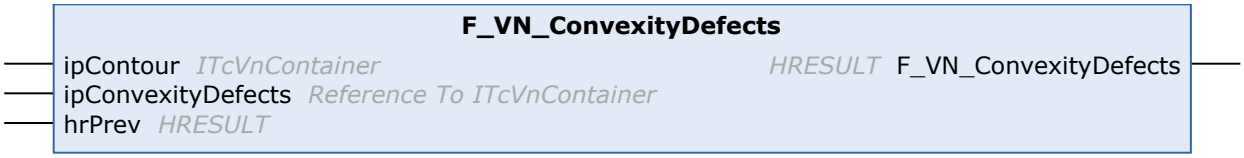
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.18 F_VN_ConvexityDefects



Finds the convexity defects of a point set.

Syntax

Definition:

```
FUNCTION F_VN_ConvexityDefects : HRESULT
VAR_INPUT
    ipContour          : ITcVnContainer;
    ipConvexityDefects : Reference To ITcVnContainer;
    hrPrev             : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContour	ITcVnContainer [▶ 385]	Source contour (ContainerType_Vector_TcVnPoint2_DINT; The elements of this container are the individual points.)
ipConvexityDefects	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the convexity defects (ContainerType_Vector_TcVnVector4_DINT with [startIndex, endIndex, furthestPointIndex, fixedPointDistance (8 fraction bits, i.e. divide by 256 to get the distance in pixels)]; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

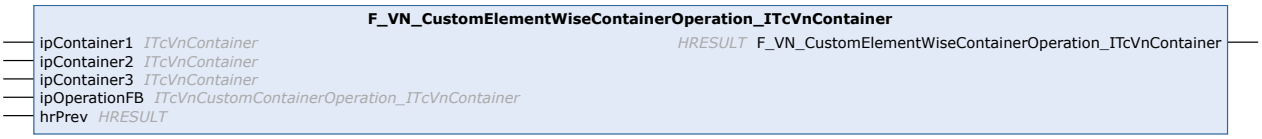
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.19 F_VN_CustomElementWiseContainerOperation_ITcVnContainer



Performs a custom, element wise operation on a set of up to 3 containers.

Syntax

Definition:

```
FUNCTION F_VN_CustomElementWiseContainerOperation_ITcVnContainer : HRESULT
VAR_INPUT
    ipContainer1  : ITcVnContainer;
    ipContainer2  : ITcVnContainer;
    ipContainer3  : ITcVnContainer;
    ipOperationFB : ITcVnCustomContainerOperation_ITcVnContainer;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer1	ITcVnContainer [▶ 385]	Container 1
ipContainer2	ITcVnContainer [▶ 385]	Container 2
ipContainer3	ITcVnContainer [▶ 385]	Container 3
ipOperationFB	ITcVnCustomContainerOperation_ITcVnContainer [▶ 259]	Custom operation on the container elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

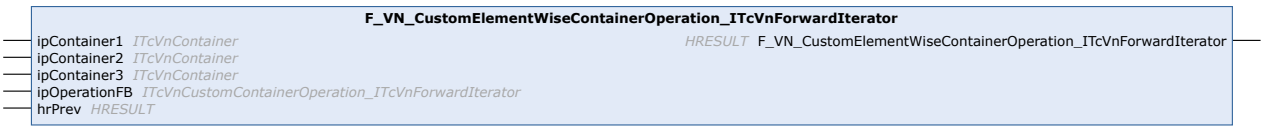
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.20 F_VN_CustomElementWiseContainerOperation_ITcVnForwardIterator



Performs a custom, element wise operation on a set of up to 3 containers.

Syntax

Definition:

```
FUNCTION F_VN_CustomElementWiseContainerOperation_ITcVnForwardIterator : HRESULT
VAR_INPUT
    ipContainer1   : ITcVnContainer;
    ipContainer2   : ITcVnContainer;
    ipContainer3   : ITcVnContainer;
    ipOperationFB : ITcVnCustomContainerOperation_ITcVnForwardIterator;
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContainer1	ITcVnContainer [▶ 385]	Container 1
ipContainer2	ITcVnContainer [▶ 385]	Container 2
ipContainer3	ITcVnContainer [▶ 385]	Container 3
ipOperationFB	ITcVnCustomContainerOperation_ITcVnForwardIterator [▶ 260]	Custom operation on the elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

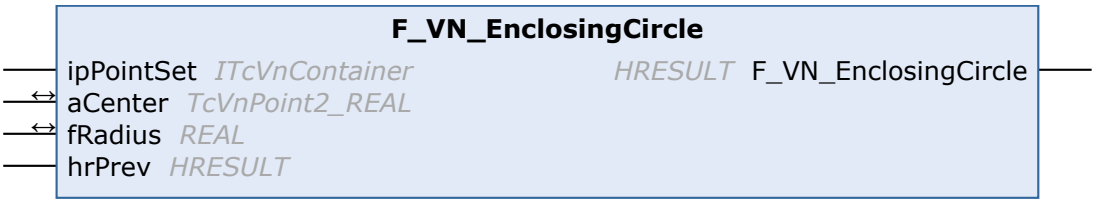
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.21 F_VN_EnclosingCircle



Searchs for a minimum area circle enclosing a set of points.

Syntax

Definition:

```
FUNCTION F_VN_EnclosingCircle : HRESULT
VAR_INPUT
    ipPointSet : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aCenter    : TcVnPoint2_REAL;
    fRadius    : REAL;
END_VAR
VAR_INPUT
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipPointSet	ITcVnContainer [▶ 385]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCenter	TcVnPoint2_REAL [▶ 171]	Returns the center of the circle
fRadius	REAL	Returns the radius of the circle

Return value

HRESULT [▶ 150]

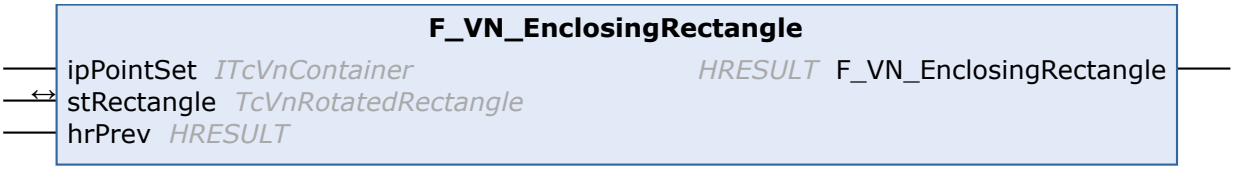
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.22 F_VN_EnclosingRectangle



Searches for a minimum area rectangle enclosing a set of points.

Syntax

Definition:

```
FUNCTION F_VN_EnclosingRectangle : HRESULT
VAR_INPUT
    ipPointSet  : ITcVnContainer;
END_VAR
VAR_IN_OUT
    stRectangle : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipPointSet	ITcVnContainer [▶ 385]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRectangle	TcVnRotatedRectangle [▶ 258]	Returns the determined rectangle

Return value

HRESULT [▶ 150]

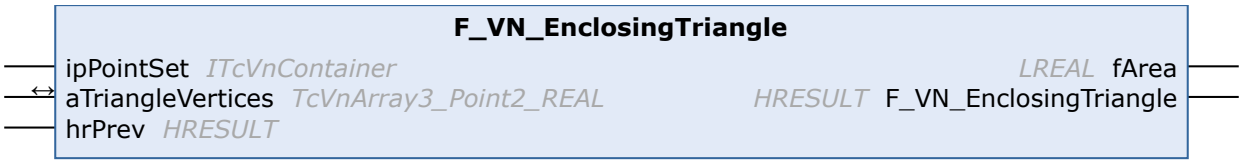
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.23 F_VN_EnclosingTriangle



Searches for a minimum area triangle enclosing a set of points.

Syntax

Definition:

```
FUNCTION F_VN_EnclosingTriangle : HRESULT
VAR_INPUT
    ipPointSet        : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aTriangleVertices : TcVnArray3_Point2_REAL;
END_VAR
VAR_INPUT
    hrPrev            : HRESULT;
END_VAR
VAR_OUTPUT
    fArea             : LREAL;
END_VAR
```

Inputs

Name	Type	Description
ipPointSet	ITcVnContainer [▶ 385]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTriangleVertices	TcVnArray3_Point2_REAL [▶ 173]	Returns the 3 triangle vertices

Outputs

Name	Type	Description
fArea	LREAL	Returns the triangle area

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.24 F_VN_FitEllipse

Fit an ellipse in a point set.

Syntax

Definition:

```
FUNCTION F_VN_FitEllipse : HRESULT
VAR_INPUT
    ipPointSet : ITcVnContainer;
END_VAR
VAR_IN_OUT
    stEllipse  : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipPointSet	ITcVnContainer [▶ 385]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; At least 5 reasonable points are required! The elements of this container are the individual points.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stEllipse	TcVnRotatedRectangle [▶ 258]	Resulting ellipse, described by a rotated rectangle.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.25 F_VN_FitLine

Fit a line into a point set.

Syntax

Definition:

```
FUNCTION F_VN_FitLine : HRESULT
VAR_INPUT
    ipPointSet : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aFitLine   : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipPointSet	ITcVnContainer [▶ 385]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aFitLine	TcVnVector4_LREAL [▶ 173]	Resulting line. The first and second element describe the x and y component of a vector collinear to the line. The third and fourth element describe the x and y component of a point on the line.

Return value

HRESULT [▶ 150]

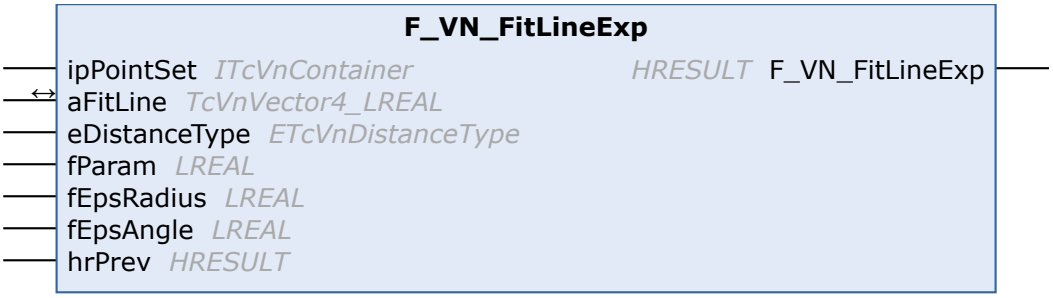
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.26 F_VN_FitLineExp



Fit a line into a point set. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FitLineExp : HRESULT
VAR_INPUT
    ipPointSet    : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aFitLine      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    eDistanceType : ETcVnDistanceType;
    fParam        : LREAL;
    fEpsRadius    : LREAL;
    fEpsAngle     : LREAL;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipPointSet	ITcVnContainer [▶ 385]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
eDistanceType	ETcVnDistanceType [▶ 207]	Distance computation method (supported: L1, L2, L12, FAIR, WELSCH, HUBER)
fParam	LREAL	Numerical parameter (c) for some eDistanceType (should be >= 0). If 0, an optimal value is chosen internally.
fEpsRadius	LREAL	Accuracy of the radius (distance of the line from the coordinate origin, should be > 0). A smaller value means higher accuracy.
fEpsAngle	LREAL	Accuracy of the angle (should be > 0). A smaller value means higher accuracy.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aFitLine	TcVnVector4_LREAL [▶ 173]	Resulting line. The first and second element describe the x and y component of a vector collinear to the line. The third and fourth element describe the x and y component of a point on the line.

Return value

HRESULT [▶ 150]

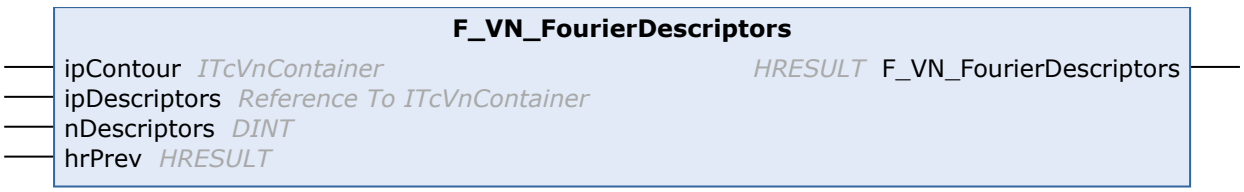
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.27 F_VN_FourierDescriptors



Computes the fourier descriptors for a closed contour.

Syntax

Definition:

```
FUNCTION F_VN_FourierDescriptors : HRESULT
VAR_INPUT
    ipContour     : ITcVnContainer;
    ipDescriptors : Reference To ITcVnContainer;
    nDescriptors  : DINT;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContour	ITcVnContainer [▶ 385]	Closed contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL. Providing the full contour is strongly recommended, i.e. use TCVN_CAM_NONE for contour detection algorithms.)
ipDescriptors	Reference To ITcVnContainer [▶ 385]	Returns the fourier descriptors for ipContour (Re0, Im0, Re1, Im1, ...; ContainerType_Vector_LREAL if ipContour is of type ContainerType_Vector_TcVnPoint2_LREAL, else ContainerType_Vector_REAL.)
nDescriptors	DINT	Specifies how many descriptors should be returned (set to -1 to return all computed descriptors, i.e. at least as many as the number of contour points)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

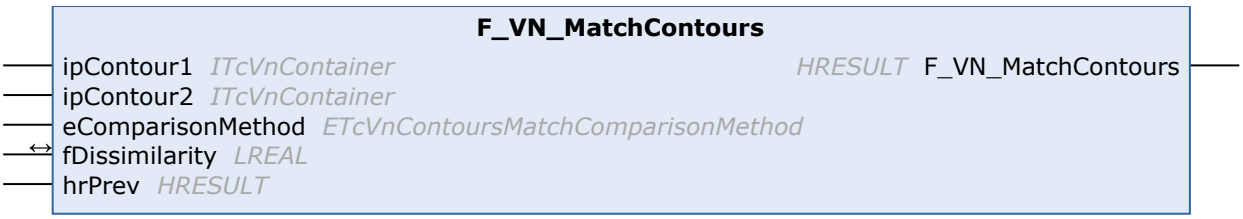
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.28 F_VN_MatchContours



Compare contours using the Hu moment invariants. In case of multiple contours in each container, the best matches are found and the average dissimilarity over all matched contours is returned.

Syntax

Definition:

```
FUNCTION F_VN_MatchContours : HRESULT
VAR_INPUT
    ipContour1        : ITcVnContainer;
    ipContour2        : ITcVnContainer;
    eComparisonMethod : ETcVnContoursMatchComparisonMethod;
END_VAR
VAR_IN_OUT
    fDissimilarity    : LREAL;
END_VAR
VAR_INPUT
    hrPrev            : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContour1	ITcVnContainer [▶ 385]	First contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour) or collection of multiple contours (ContainerType_Vector_Vector_TcVnPoint2_DINT)
ipContour2	ITcVnContainer [▶ 385]	Second contour (same type as ipContour1)
eComparisonMethod	ETcVnContoursMatchComparisonMethod [▶ 204]	Method used for comparing the Hu moment invariants of the contours
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fDissimilarity	LREAL	Returns the dissimilarity of the contours depending on the chosen comparison method

Return value

HRESULT [▶ 150]

Beispiele

- Match Contours 1vsN (manuelle Formen) [▶ 2857]
- Match Contours (extrahierte Formen) [▶ 2861]

Verwandte Funktionen

- F_VN_ContourMoments [▶ 996]
- F_VN_HuMomentInvariants [▶ 1569]
- F_VN_MatchContours [▶ 1013]
- F_VN_MatchContours1vsN [▶ 1014]

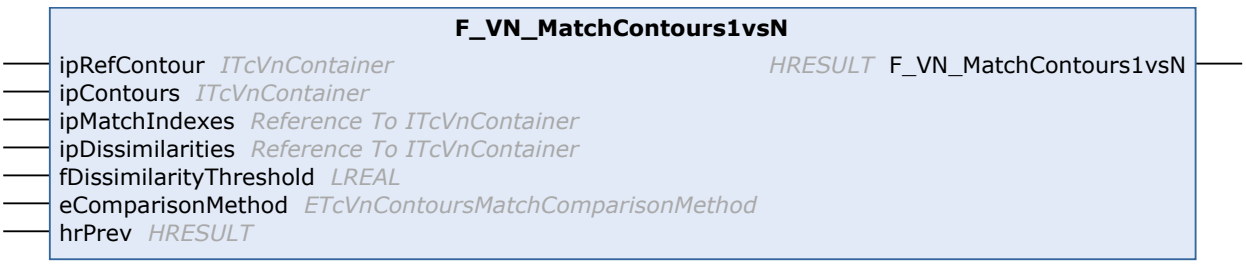
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.29 F_VN_MatchContours1vsN



Compare a reference contour with multiple other contours using the Hu moment invariants. Returns a sorted list of best matches.

Syntax

Definition:

```
FUNCTION F_VN_MatchContours1vsN : HRESULT
VAR_INPUT
    ipRefContour            : ITcVnContainer;
    ipContours              : ITcVnContainer;
    ipMatchIndexes          : Reference To ITcVnContainer;
    ipDissimilarities       : Reference To ITcVnContainer;
    fDissimilarityThreshold : LREAL;
    eComparisonMethod       : ETcVnContoursMatchComparisonMethod;
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipRefContour	ITcVnContainer [▶ 385]	Reference contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour)
ipContours	ITcVnContainer [▶ 385]	Collection of multiple contours (CTcVnContainer_Vector_Vector_TcVnPoint2_DINT)
ipMatchIndexes	Reference To ITcVnContainer [▶ 385]	Returns the indexes of the best matches (CTcVnContainer_Vector_ULINT; sorted, first element is best match, i.e. has lowest dissimilarity)
ipDissimilarities	Reference To ITcVnContainer [▶ 385]	Returns the computed dissimilarities of the best matches (CTcVnContainer_Vector_LREAL; Optional, set to 0 if not required; sorted corresponding to ipMatchIndexes)
fDissimilarityThreshold	LREAL	Neglect irrelevant matches, i.e. dissimilarity > fDissimilarityThreshold
eComparisonMethod	ETcVnContoursMatchComparisonMethod [▶ 204]	Method used for comparing the Hu moment invariants of the contours
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

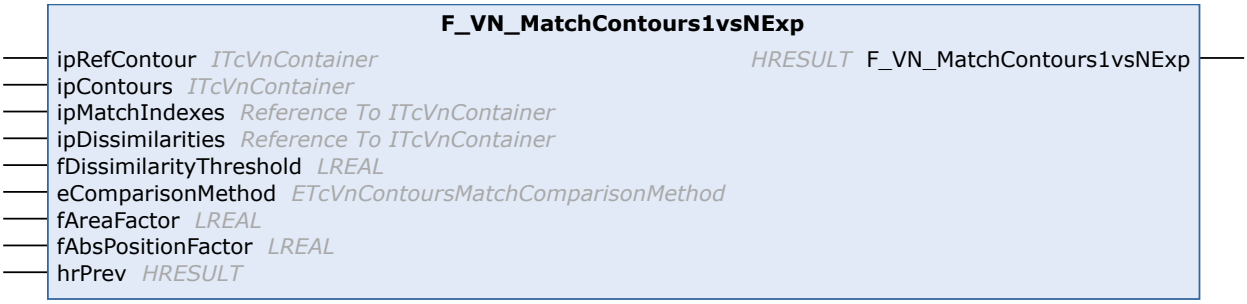
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.30 F_VN_MatchContours1vsNExp



Compare a reference contour with multiple other contours using the Hu moment invariants (and optionally further aspects). Returns a sorted list of best matches. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_MatchContours1vsNExp : HRESULT
VAR_INPUT
    ipRefContour            : ITcVnContainer;
    ipContours              : ITcVnContainer;
    ipMatchIndexes          : Reference To ITcVnContainer;
    ipDissimilarities       : Reference To ITcVnContainer;
    fDissimilarityThreshold : LREAL;
    eComparisonMethod       : ETcVnContoursMatchComparisonMethod;
    fAreaFactor             : LREAL;
    fAbsPositionFactor      : LREAL;
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipRefContour	ITcVnContainer [▶ 385]	Reference contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour)
ipContours	ITcVnContainer [▶ 385]	Collection of multiple contours (CTcVnContainer_Vector_Vector_TcVnPoint2_DINT)
ipMatchIndexes	Reference To ITcVnContainer [▶ 385]	Returns the indexes of the best matches (CTcVnContainer_Vector_ULINT; sorted, first element is best match, i.e. has lowest dissimilarity)
ipDissimilarities	Reference To ITcVnContainer [▶ 385]	Returns the computed dissimilarities of the best matches (CTcVnContainer_Vector_LREAL; Optional, set to 0 if not required; sorted corresponding to ipMatchIndexes)
fDissimilarityThreshold	LREAL	Neglect irrelevant matches, i.e. dissimilarity > fDissimilarityThreshold
eComparisonMethod	ETcVnContoursMatchComparisonMethod [▶ 204]	Method used for comparing the Hu moment invariants of the contours
fAreaFactor	LREAL	If > 0, the relative area difference between contours is scaled by this factor and added to the computed dissimilarity
fAbsPositionFactor	LREAL	If > 0, the absolute position difference between contours (i.e. the coordinates of the geometric contour centers) is scaled by this factor and added to the computed dissimilarity
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

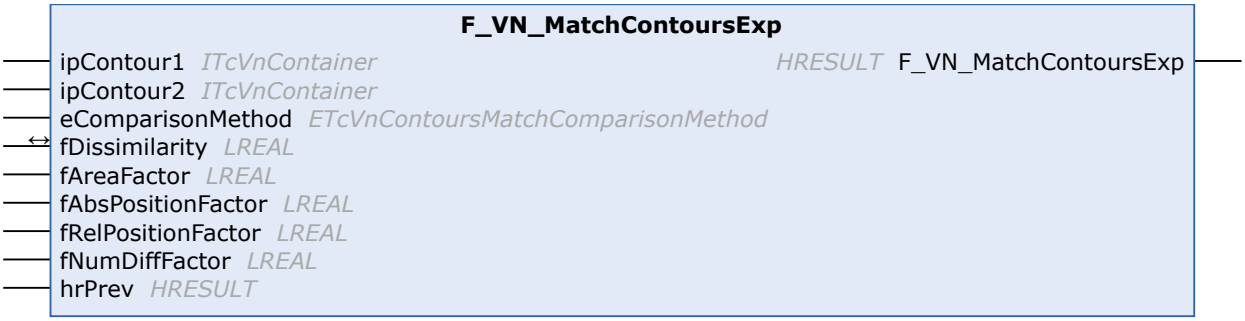
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.31 F_VN_MatchContoursExp



Compare contours using the Hu moment invariants (and optionally further aspects). In case of multiple contours in each container, the best matches are found and the average dissimilarity over all matched contours is returned. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_MatchContoursExp : HRESULT
VAR_INPUT
    ipContour1          : ITcVnContainer;
    ipContour2          : ITcVnContainer;
    eComparisonMethod   : ETcVnContoursMatchComparisonMethod;
END_VAR
VAR_IN_OUT
    fDissimilarity      : LREAL;
END_VAR
VAR_INPUT
    fAreaFactor         : LREAL;
    fAbsPositionFactor  : LREAL;
    fRelPositionFactor  : LREAL;
    fNumDiffFactor      : LREAL;
    hrPrev              : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContour1	ITcVnContainer [▶ 385]	First contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour) or collection of multiple contours (ContainerType_Vector_Vector_TcVnPoint2_DINT)
ipContour2	ITcVnContainer [▶ 385]	Second contour (same type as ipContour1)
eComparisonMethod	ETcVnContoursMatchComparisonMethod [▶ 204]	Method used for comparing the Hu moment invariants of the contours
fAreaFactor	LREAL	If > 0, the relative area difference between contours is scaled by this factor and added to the computed dissimilarity
fAbsPositionFactor	LREAL	If > 0, the absolute position difference between contours (i.e. the coordinates of the geometric contour centers) is scaled by this factor and added to the computed dissimilarity
fRelPositionFactor	LREAL	If > 0, the relative position difference between contours is scaled by this factor and added to the computed dissimilarity (only used if matching multiple contours)
fNumDiffFactor	LREAL	If > 0, the difference between the number of contours in both containers is scaled by this factor and added to the computed dissimilarity (only used if matching multiple contours)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fDissimilarity	LREAL	Returns the dissimilarity of the contours depending on the chosen comparison method

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.32 F_VN_UprightBoundingRectangle

Determines the upright bounding rectangle of a set of points.

Syntax

Definition:

```
FUNCTION F_VN_UprightBoundingRectangle : HRESULT
VAR_INPUT
    ipPointSet  : ITcVnContainer;
END_VAR
VAR_IN_OUT
    stRectangle : TcVnRectangle_UDINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipPointSet	ITcVnContainer [▶ 385]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRectangle	TcVnRectangle_UDINT [▶ 258]	Returns the determined rectangle

Return value

HRESULT [▶ 150]

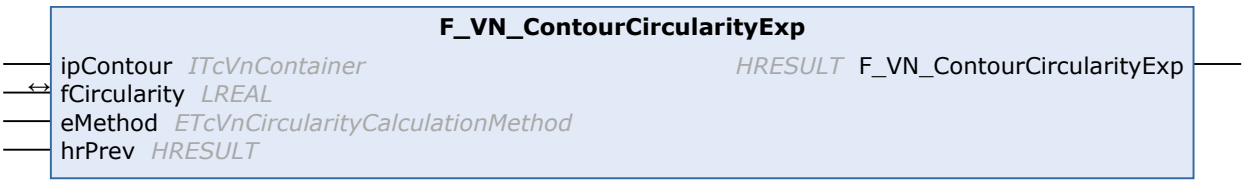
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.9.33 F_VN_ContourCircularityExp



Computes the circularity of a contour (1.0: ideal circle, 0.0: straight line). (expert function)

Syntax

Definition:

```
FUNCTION F_VN_ContourCircularityExp : HRESULT
VAR_INPUT
    ipContour    : ITcVnContainer;
END_VAR
```

```
VAR_IN_OUT
    fCircularity : LREAL;
END_VAR
VAR_INPUT
    eMethod      : ETcVnCircularityCalculationMethod;
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContour	ITcVnContainer [▶ 385]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
eMethod	ETcVnCircularityCalculationMethod [▶ 237]	Method used for computing the circularity
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fCircularity	LREAL	Returns the circularity of the contour [0..1]

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.10 Control Functions

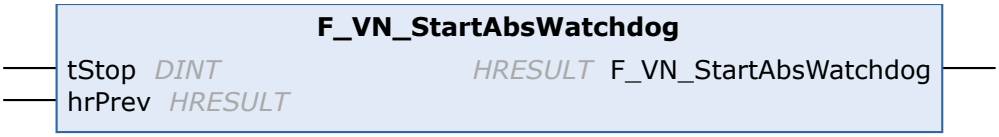
Diese Gruppe enthält Funktionen zur Ablaufsteuerung, z. B. zur Handhabung von Watchdogs [▶ 156].

Funktionen

Watchdogs

- F_VN_StartAbsWatchdog(Exp) [▶ 1021]
- F_VN_StartRelWatchdog(Exp) [▶ 1023]
- F_VN_StopWatchdog [▶ 1024]

6.1.4.10.1 F_VN_StartAbsWatchdog



Starts a cooperative watchdog given an absolute stop time.

Syntax

Definition:

```
FUNCTION F_VN_StartAbsWatchdog : HRESULT
VAR_INPUT
    tStop  : DINT;
    hrPrev : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
tStop	DINT	Stop time in us
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Weitere Informationen finden Sie im Kapitel Watchdogs [▶ 156].

tStop-Zeit überschritten

Beachten Sie, dass die tStop-Zeit bei Auslösung des Watchdogs geringfügig (einige µs) überschritten wird.

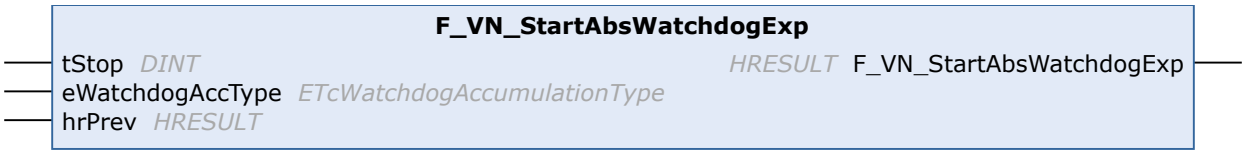
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.10.2 F_VN_StartAbsWatchdogExp



Start a cooperative watchdog given an absolute stop time. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_StartAbsWatchdogExp : HRESULT
VAR_INPUT
    tStop            : DINT;
    eWatchdogAccType : ETcWatchdogAccumulationType;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
tStop	DINT	Stop time in us
eWatchdogAccType	ETcWatchdogAccumulationType [▶ 235]	Accumalation method used for combining the results of multiple functions enclosed by this watchdog
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

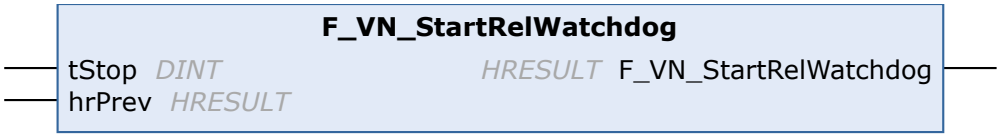
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.10.3 F_VN_StartRelWatchdog



Starts a cooperative watchdog given a stop time relative to the current time.

Syntax

Definition:

```
FUNCTION F_VN_StartRelWatchdog : HRESULT
VAR_INPUT
    tStop  : DINT;
    hrPrev : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
tStop	DINT	Stop time in us
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Weitere Informationen finden Sie im Kapitel Watchdogs [▶ 156].

tStop-Zeit überschritten

Beachten Sie, dass die tStop-Zeit bei Auslösung des Watchdogs geringfügig (einige µs) überschritten wird.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.10.4 F_VN_StartRelWatchdogExp

Start a cooperative watchdog given a stop time relative to the current time. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_StartRelWatchdogExp : HRESULT
VAR_INPUT
    tStop            : DINT;
    eWatchdogAccType : ETcWatchdogAccumulationType;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
tStop	DINT	Stop time in us
eWatchdogAccType	ETcWatchdogAccumulationType [▶ 235]	Accumalation method used for combining the results of multiple functions enclosed by this watchdog
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

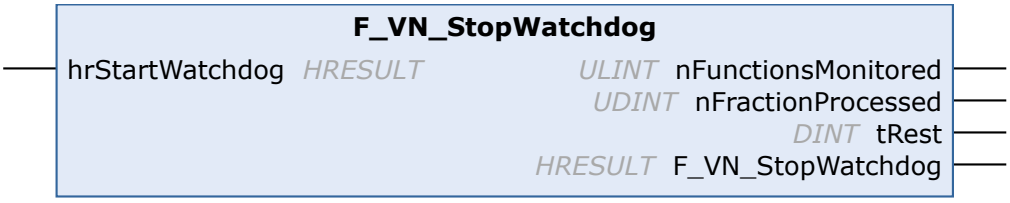
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.10.5 F_VN_StopWatchdog



Stops a watchdog and provide runtime information.

Syntax

Definition:

```
FUNCTION F_VN_StopWatchdog : HRESULT
VAR_INPUT
    hrStartWatchdog     : HRESULT;
END_VAR
VAR_OUTPUT
    nFunctionsMonitored : ULINT;
    nFractionProcessed  : UDINT;
    tRest               : DINT;
END_VAR
```

Inputs

Name	Type	Description
hrStartWatchdog	HRESULT [▶ 150]	HRESULT indicating the result of the function used to start the watchdog (If SUCCEEDED(hrStartWatchdog) equals false, no operation is executed.)

Outputs

Name	Type	Description
nFunctionsMonitored	ULINT	Returns the number of functions monitored
nFractionProcessed	UDINT	Returns the fraction processed accumulated over the monitored functions in percent
tRest	DINT	Returns the remaining computation time in us (may be negative)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Der Output nFractionProcessed muss je nach Parametrierung des Watchdogs anders interpretiert werden. Dies wird mit dem ETcWatchdogAccumulationType [▶ 235] beim Starten des Watchdogs eingestellt.

Weitere Informationen finden Sie unter Watchdogs [▶ 156].

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11 Drawing

Diese Gruppe enthält Funktionen zum Bemalen und Beschreiben von Bildern.

Handhabung von 4-kanaligen Farben

4-kanalige Farben werden direkt in das Ergebnisbild übernommen. Es wird kein Alpha-Blending o. Ä. betrieben.

Funktionen

Arrow

- F_VN_DrawArrow(Exp) [▶ 1026]
- F_VN_DrawArrow(Exp)_TcVnVector4_DINT [▶ 1027]

Circles und Arcs

- F_VN_DrawCircle(Exp) [▶ 1031]
- F_VN_DrawCircles(Exp) [▶ 1033]
- F_VN_DrawCircularArc(Exp) [▶ 1035]
- F_VN_FillCircle [▶ 1074]

Components

- F_VN_DrawComponents(Exp) [▶ 1037]

Contours

- F_VN_DrawContours(Exp) [▶ 1040]
- F_VN_FillContours [▶ 1075]

Ellipse

- F_VN_DrawEllipse(Exp) [▶ 1043]
- F_VN_FillEllipse [▶ 1076]

Keypoints und Matches

- F_VN_DrawKeypoints(Exp) [▶ 1045]
- F_VN_DrawMatches(Exp) [▶ 1057]

Lines

- F_VN_DrawLine(Exp) [▶ 1047]
- F_VN_DrawLines(Exp) [▶ 1054]
- F_VN_DrawLine(Exp)_TcVnVector4_DINT [▶ 1048]
- F_VN_DrawLine(Exp)_TcVnVector4_LREAL [▶ 1050]

Orientation

- F_VN_DrawOrientation(Exp) [▶ 1059]

Points

- F_VN_DrawPoint(Exp) [▶ 1062]
- F_VN_DrawPoints(Exp) [▶ 1064]

Polygon

- F_VN_DrawPolygon(Exp) [▶ 1066]
- F_VN_FillPolygon [▶ 1077]

Rectangle

- F_VN_DrawRectangle [▶ 1068]
- F_VN_DrawRectangle_TcVnRectangle_DINT [▶ 1069]
- F_VN_DrawRectangle_TcVnRectangle_UDINT [▶ 1070]
- F_VN_DrawRotatedRectangle(Exp) [▶ 1071]
- F_VN_FillRectangle [▶ 1078]
- F_VN_FillRotatedRectangle [▶ 1079]

Intensity Profile

- F_VN_PlotIntensityProfile(Exp) [▶ 1080]

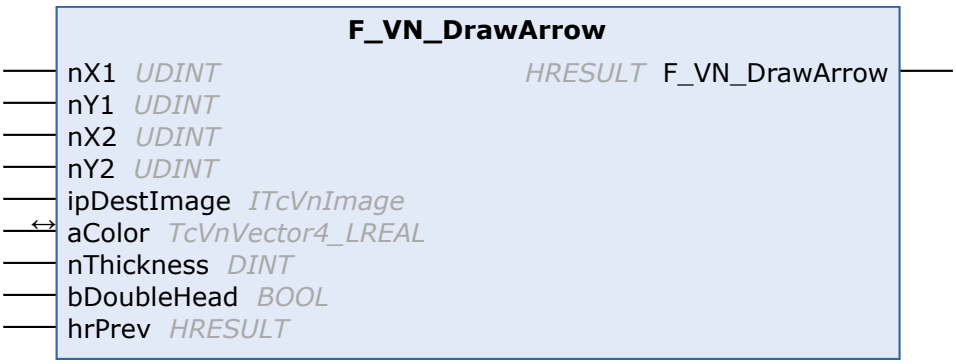
Text

- F_VN_PutLabel(Exp) [▶ 1082]
- F_VN_PutText(Exp) [▶ 1084]

Malen und Füllen

Für viele Formen gibt es die Möglichkeit, sowohl den Umriss der Form, als auch die gefüllte Form zu malen. Dazu stehen je zwei Funktion zur Verfügung (`F_VN_Draw…` und `F_VN_Fill…`). Außerdem führt ein Setzen der Umrissdicke `nThickness` auf `-1` auch dazu, dass die Form gefüllt wird.

6.1.4.11.1 F_VN_DrawArrow



Draw an arrowed line.

Syntax

Definition:

```
FUNCTION F_VN_DrawArrow : HRESULT
VAR_INPUT
    nX1          : UDINT;
    nY1          : UDINT;
    nX2          : UDINT;
    nY2          : UDINT;
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor       : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness   : DINT;
    bDoubleHead : BOOL;
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nX1	UDINT	x coordinate of the start point
nY1	UDINT	y coordinate of the start point
nX2	UDINT	x coordinate of the end point
nY2	UDINT	y coordinate of the end point
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness
bDoubleHead	BOOL	To draw an arrow on each side of the line.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

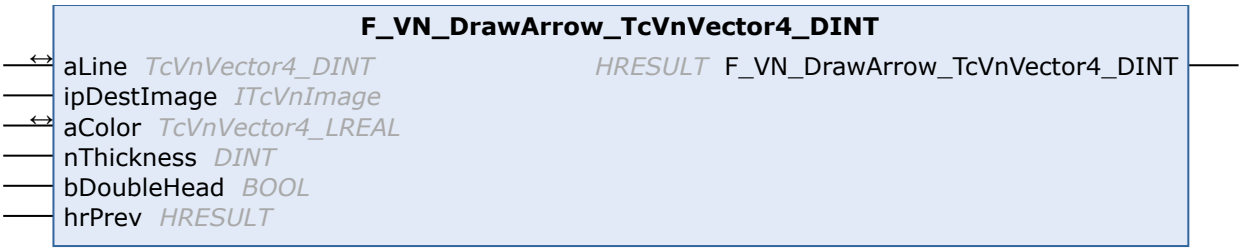
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.2 F_VN_DrawArrow_TcVnVector4_DINT



Draw an arrowed line.

Syntax

Definition:

```
FUNCTION F_VN_DrawArrow_TcVnVector4_DINT : HRESULT
VAR_IN_OUT
    aLine       : TcVnVector4_DINT;
END_VAR
VAR_INPUT
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
```

```
VAR_INPUT
    nThickness  : DINT;
    bDoubleHead : BOOL;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness
bDoubleHead	BOOL	To draw an arrow on each side of the line.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLine	TcVnVector4_DINT [▶ 173]	The start and end point of a line segment [x1, y1, x2, y2]
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

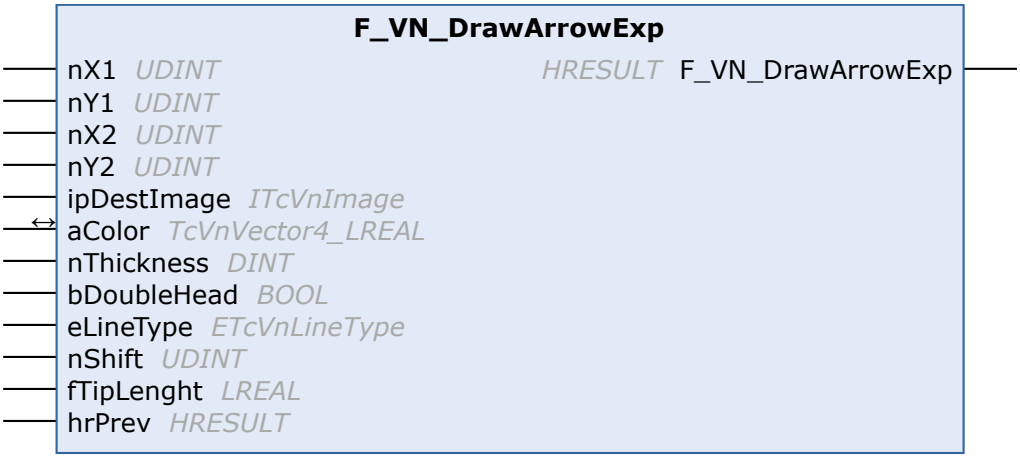
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.3 F_VN_DrawArrowExp



Draws an arrowed line. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_DrawArrowExp : HRESULT
VAR_INPUT
    nX1         : UDINT;
    nY1         : UDINT;
    nX2         : UDINT;
    nY2         : UDINT;
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness  : DINT;
    bDoubleHead : BOOL;
    eLineType   : ETcVnLineType;
    nShift      : UDINT;
    fTipLenght  : LREAL;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nX1	UDINT	x coordinate of the start point
nY1	UDINT	y coordinate of the start point
nX2	UDINT	x coordinate of the end point
nY2	UDINT	y coordinate of the end point
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness
bDoubleHead	BOOL	To draw an arrow on each side of the line.
eLineType	ETcVnLineType [▶ 219]	Line type
nShift	UDINT	Fractional bits of the coordinates (bit shift)
fTipLenght	LREAL	Lenght of the arrow's tip as a fraction of the line's lenght
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

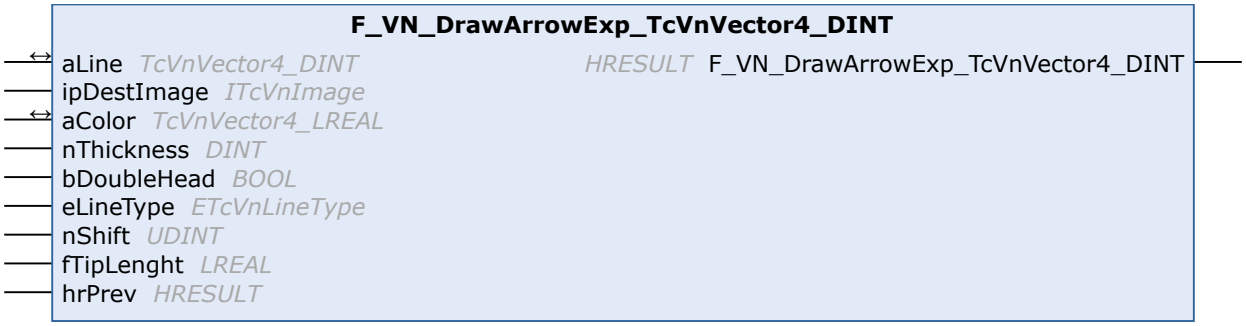
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.4 F_VN_DrawArrowExp_TcVnVector4_DINT



Draws an arrowed line. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_DrawArrowExp_TcVnVector4_DINT : HRESULT
VAR_IN_OUT
    aLine       : TcVnVector4_DINT;
END_VAR
VAR_INPUT
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness  : DINT;
    bDoubleHead : BOOL;
    eLineType   : ETcVnLineType;
    nShift      : UDINT;
    fTipLenght  : LREAL;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness
bDoubleHead	BOOL	To draw an arrow on each side of the line.
eLineType	ETcVnLineType [▶ 219]	Line type
nShift	UDINT	Fractional bits of the coordinates (bit shift)
fTipLenght	LREAL	Lenght of the arrow's tip as a fraction of the line's lenght
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLine	TcVnVector4_DINT [▶ 173]	The start and end point of a line segment [x1, y1, x2, y2]
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

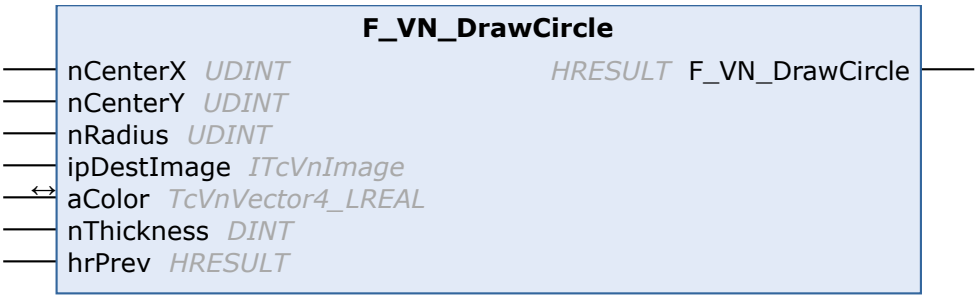
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.5 F_VN_DrawCircle



Draws a circle.

Syntax

Definition:

```
FUNCTION F_VN_DrawCircle : HRESULT
VAR_INPUT
    nCenterX    : UDINT;
    nCenterY    : UDINT;
    nRadius     : UDINT;
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness  : DINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nCenterX	UDINT	x coordinate of the center
nCenterY	UDINT	y coordinate of the center
nRadius	UDINT	Radius
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness (if negative, the circle is filled)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

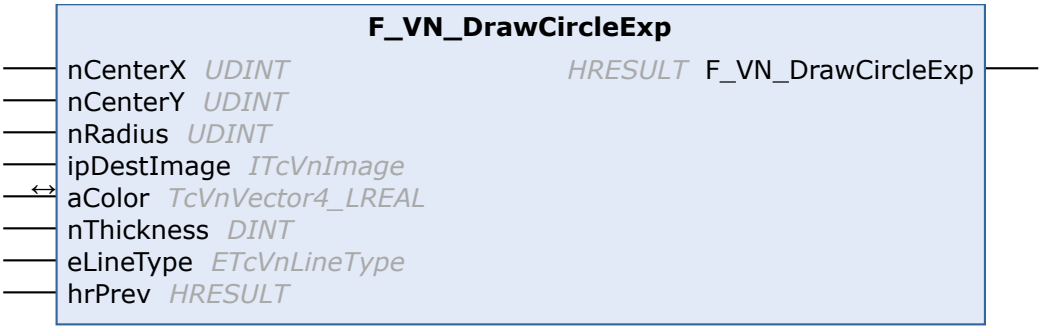
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.6 F_VN_DrawCircleExp



Draws a circle.

Syntax

Definition:

```
FUNCTION F_VN_DrawCircleExp : HRESULT
VAR_INPUT
    nCenterX    : UDINT;
    nCenterY    : UDINT;
    nRadius     : UDINT;
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness  : DINT;
    eLineType   : ETcVnLineType;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nCenterX	UDINT	x coordinate of the center
nCenterY	UDINT	y coordinate of the center
nRadius	UDINT	Radius
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness (if negative, the circle is filled)
eLineType	ETcVnLineType [▶ 219]	Line type
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

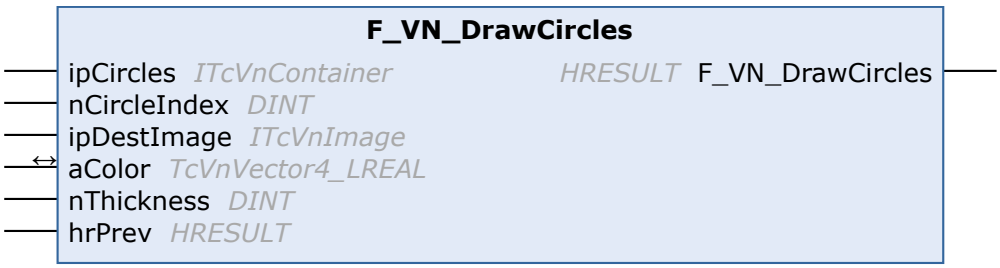
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.7 F_VN_DrawCircles



Draws circles.

Syntax

Definition:

```
FUNCTION F_VN_DrawCircles : HRESULT
VAR_INPUT
    ipCircles    : ITcVnContainer;
    nCircleIndex : DINT;
    ipDestImage  : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor       : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness   : DINT;
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipCircles	ITcVnContainer [▶ 385]	Container with circles (ContainerType_Vector_TcVnVector3_REAL; Each container element contains the x coordinate of the circle center [0], the y coordinate of the circle center [1], and the radius [2].)
nCircleIndex	DINT	Index of a specific circle to be drawn (if negative, all circles within the container are drawn)
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness (if negative, the circle is filled)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

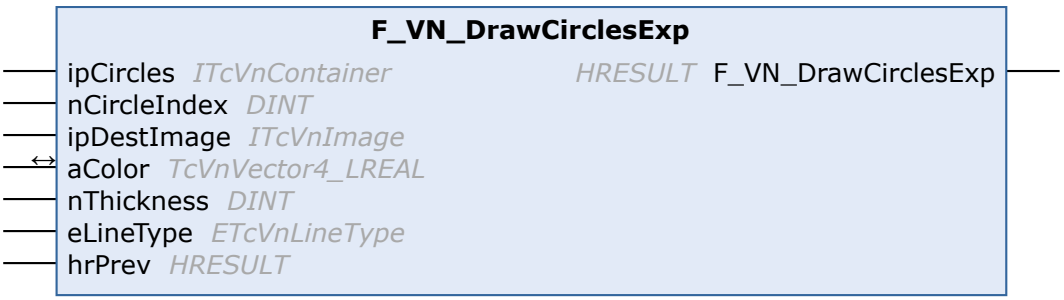
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.8 F_VN_DrawCirclesExp



Draws circles. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_DrawCirclesExp : HRESULT
VAR_INPUT
    ipCircles    : ITcVnContainer;
    nCircleIndex : DINT;
    ipDestImage  : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor       : TcVnVector4_LREAL;
```

```
END_VAR
VAR_INPUT
    nThickness    : DINT;
    eLineType     : ETcVnLineType;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipCircles	ITcVnContainer [▶ 385]	Container with circles (ContainerType_Vector_TcVnVector3_REAL; Each container element contains the x coordinate of the circle center [0], the y coordinate of the circle center [1], and the radius [2].)
nCircleIndex	DINT	Index of a specific circle to be drawn (if negative, all circles within the container are drawn)
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness (if negative, the circle is filled)
eLineType	ETcVnLineType [▶ 219]	Line type
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

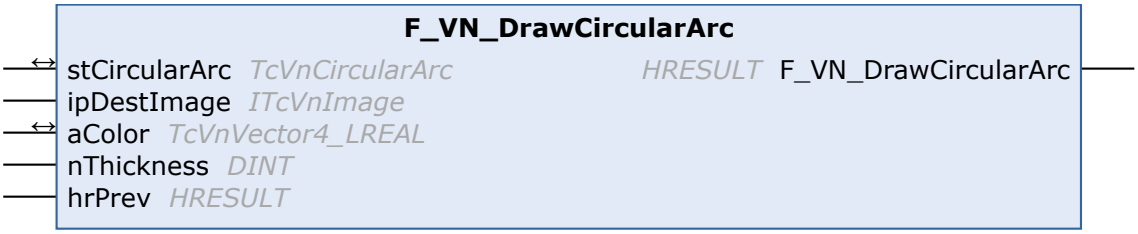
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.9 F_VN_DrawCircularArc



Draws a circular arc.

Syntax

Definition:

```
FUNCTION F_VN_DrawCircularArc : HRESULT
VAR_IN_OUT
    stCircularArc : TcVnCircularArc;
END_VAR
VAR_INPUT
    ipDestImage   : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor        : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness    : DINT;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness (if negative, the arc is filled)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stCircularArc	TcVnCircularArc [▶ 241]	Circular arc definition
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

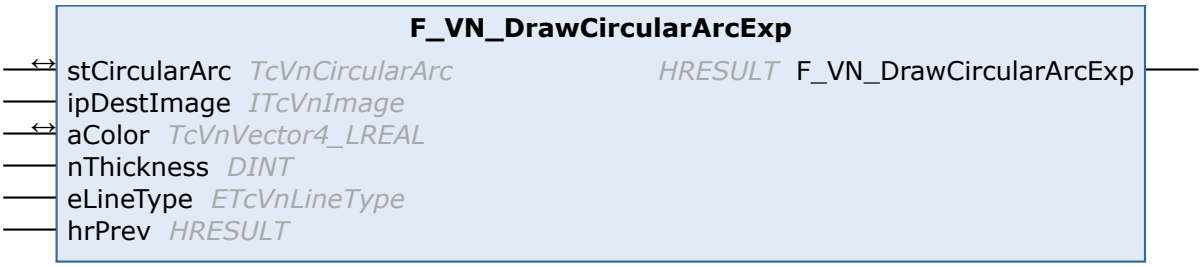
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.10 F_VN_DrawCircularArcExp



Draws a circular arc. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_DrawCircularArcExp : HRESULT
VAR_IN_OUT
    stCircularArc : TcVnCircularArc;
END_VAR
VAR_INPUT
    ipDestImage   : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor        : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness    : DINT;
    eLineType     : ETcVnLineType;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness (if negative, the arc is filled)
eLineType	ETcVnLineType [▶ 219]	Line type
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stCircularArc	TcVnCircularArc [▶ 241]	Circular arc definition
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

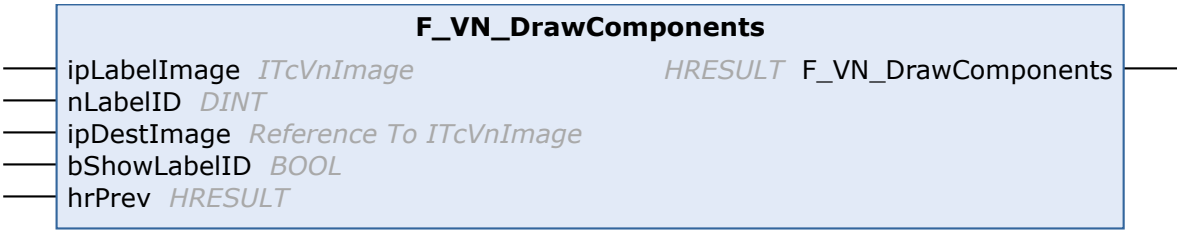
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.11 F_VN_DrawComponents



Draw the results of connected component function.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_DrawComponents : HRESULT
VAR_INPUT
    ipLabelImage : ITcVnImage;
    nLabelID     : DINT;
    ipDestImage  : Reference To ITcVnImage;
    bShowLabelID : BOOL;
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipLabelImage	ITcVnImage [▶ 426]	Image contains labels for each pixel (1 channel DINT or UINT).
nLabelID	DINT	Value of a specific label to be drawn (if negative, all components within the image are drawn)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate image will be created if required. The output image is 1 or 3 channels USINT).
bShowLabelID	BOOL	Show label ID in the center of the component on the destination image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

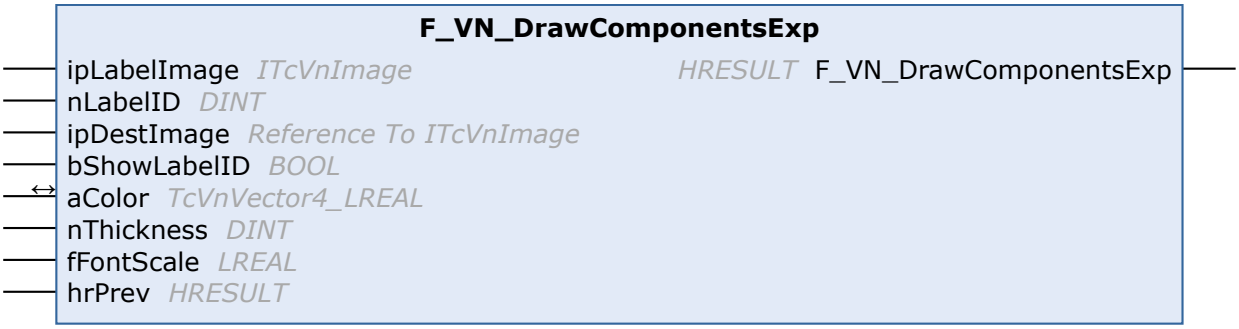
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.12 F_VN_DrawComponentsExp



Draw the results of connected component function. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_DrawComponentsExp : HRESULT
VAR_INPUT
    ipLabelImage : ITcVnImage;
    nLabelID     : DINT;
    ipDestImage  : Reference To ITcVnImage;
    bShowLabelID : BOOL;
END_VAR
VAR_IN_OUT
    aColor       : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness   : DINT;
    fFontScale   : LREAL;
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipLabelImage	ITcVnImage [▶ 426]	Image contains labels for each pixel (1 channel DINT or UINT).
nLabelID	DINT	Value of a specific label to be drawn (if negative, all components within the image are drawn)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate image will be created if required. The output image is 1 or 3 channels USINT).
bShowLabelID	BOOL	Show lable ID in the center of the component on the destination image
nThickness	DINT	Text line thickness
fFontScale	LREAL	Scaling factor for the text
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color to draw the components (for {-1, -1, -1, -1}, a random color is chosen for each component)

Return value

HRESULT [▶ 150]

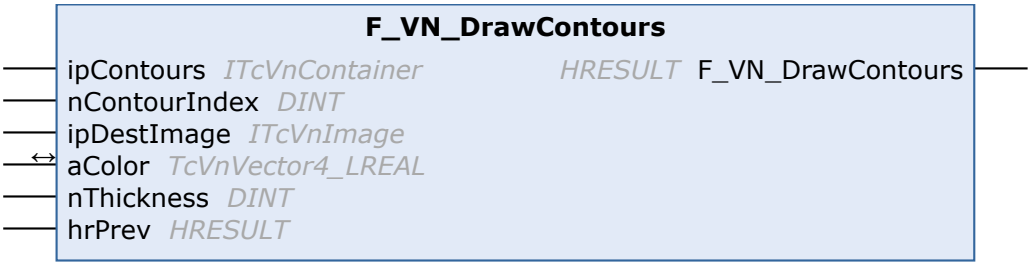
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.13 F_VN_DrawContours



Draws a single point set or multiple point sets that are interpreted as contours.

Syntax

Definition:

```
FUNCTION F_VN_DrawContours : HRESULT
VAR_INPUT
    ipContours    : ITcVnContainer;
    nContourIndex : DINT;
    ipDestImage   : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor        : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness    : DINT;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContours	ITcVnContainer [▶ 385]	Single contour (ContainerType_Vector_TcVnPoint2_DINT) or multiple contours (ContainerType_Vector_Vector_TcVnPoint2_DINT)
nContourIndex	DINT	Index of a specific contour to be drawn (if negative, all contours within the container are drawn)
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness (if negative, the contours are filled)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_DrawContours` zeichnet Konturen in das Zielbild. Eine Kontur wird dabei aus einer Menge an Punkten beschrieben, welche in einem Container gespeichert sind. Dabei ist zu beachten, dass nur ganzzahlige Konturpunkte gezeichnet werden können, sodass der Container vom Typ *ContainerType_Vector_TcVnPoint2_DINT* sein muss. Werden die Punkte über Fließkommazahlen beschrieben, so kann mit F_VN_ConvertContainerType [▶ 771] eine Konvertierung der Punkte erfolgen.

Neben einem Container mit einer Kontur kann der Funktion auch ein Container mit mehreren Konturen übergeben werden. Dieser ist dann vom Typ *ContainerType_Vector_Vector_TcVnPoint2_DINT*. Welche Kontur gezeichnet werden soll, wird über die Eingangsvariable *nContourIndex* angegeben. Sollen alle Konturen gezeichnet werden, muss *nContourIndex* auf einen negativen Wert gesetzt werden. Es ist effizienter, einen ganzen Container von Konturen auf einmal zu zeichnen (setzen von *nContourIndex* auf einen negativen Wert), als sie nacheinander zu zeichnen, indem Sie die Funktion wiederholt innerhalb einer Schleife aufrufen.

Expert-Parameter

Die Expert-Variante F_VN_DrawContoursExp [▶ 1041] enthält zusätzliche Parameter.

Anwendung

Das Malen von allen Konturen einer Kontur-Liste `ipContours` mit einer roten, 5px starken Linie sieht z. B. so aus:

```
hr := F_VN_DrawContours(
    ipContours      :=  ipContours,
    nContourIndex   :=  -1,
    ipDestImage     :=  ipImageRes,
    aColor          :=  aColorRed,
    nThickness      :=  5,
    hrPrev          :=  hr
);
```

Typische Funktionen die Konturen vom Typ ContainerType_Vector_Vector_TcVnPoint2_DINT zurückgeben

- F_VN_DetectBlobs [▶ 1208]
- F_VN_FindContours [▶ 1217] bzw. F_VN_FindContoursExp [▶ 1220]
- F_VN_FindContourHierarchyExp [▶ 1213]

Beispiele

- Blob Detection mit Watchdog Überwachung [▶ 2867]

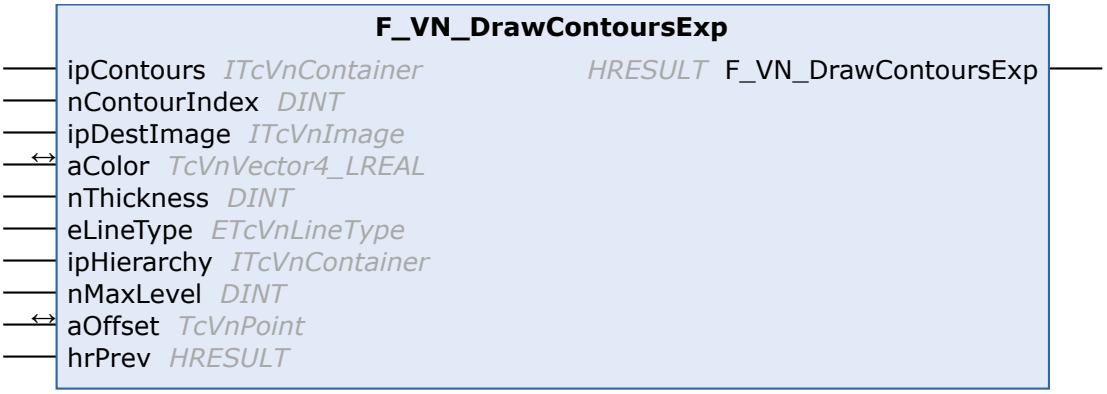
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.14 F_VN_DrawContoursExp



Draws a single point set or multiple point sets that are interpreted as contours. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_DrawContoursExp : HRESULT
VAR_INPUT
    ipContours    : ITcVnContainer;
    nContourIndex : DINT;
    ipDestImage   : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor        : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness    : DINT;
    eLineType     : ETcVnLineType;
    ipHierarchy   : ITcVnContainer;
    nMaxLevel     : DINT;
END_VAR
VAR_IN_OUT
    aOffset       : TcVnPoint;
END_VAR
VAR_INPUT
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContours	ITcVnContainer [▶ 385]	Single contour (ContainerType_Vector_TcVnPoint2_DINT) or multiple contours (ContainerType_Vector_Vector_TcVnPoint2_DINT)
nContourIndex	DINT	Index of a specific contour to be drawn (if negative, all contours within the container are drawn)
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness (if negative, the contours are filled)
eLineType	ETcVnLineType [▶ 219]	Line type
ipHierarchy	ITcVnContainer [▶ 385]	Contour hierarchy (ContainerType_Vector_TcVnVector4_DINT)
nMaxLevel	DINT	Maximum level of contours to be drawn
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color
aOffset	TcVnPoint [▶ 171]	Offset by which every contour point is shifted

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion F_VN_DrawContoursExp entspricht der Funktion F_VN_DrawContours [▶ 1040], erweitert um zusätzliche Eingangsvariablen.

So kann die Hierarchie-Beschreibung, welche neben den gefundenen Konturen von der Funktion F_VN_FindContourHierarchyExp [▶ 1212] zurückgegeben wird, mit berücksichtigt werden. Hierzu wird die Hierarchie-Beschreibung über `ipHierarchy` und über die Eingangsvariable `nMaxLevel` angegeben, bis zu welcher Hierarchy-Ebene die Konturen gezeichnet werden sollen. Wird bei `ipHierarchy` 0 übergeben, so wird die Hierarchy ignoriert.

Über `aOffset` können alle Konturpunkte in x und y Richtung einheitlich versetzt eingezeichnet werden. Dies ist zum Beispiel interessant, wenn die Konturen in einer ROI gesucht wurden, aber ins Originalbild eingezeichnet werden sollen.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.15 F_VN_DrawEllipse

Draws an ellipse.

Syntax

Definition:

```
FUNCTION F_VN_DrawEllipse : HRESULT
VAR_IN_OUT
    stEllipse   : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness  : DINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness (if negative, the ellipse is filled)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stEllipse	TcVnRotatedRectangle [▶ 258]	Ellipse to be drawn (rotation angle in degrees)
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.16 F_VN_DrawEllipseExp

Draws an ellipse. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_DrawEllipseExp : HRESULT
VAR_IN_OUT
    stEllipse   : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness  : DINT;
    eLineType   : ETcVnLineType;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness (if negative, the ellipse is filled)
eLineType	ETcVnLineType [▶ 219]	Line type
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stEllipse	TcVnRotatedRectangle [▶ 258]	Ellipse to be drawn (rotation angle in degrees)
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

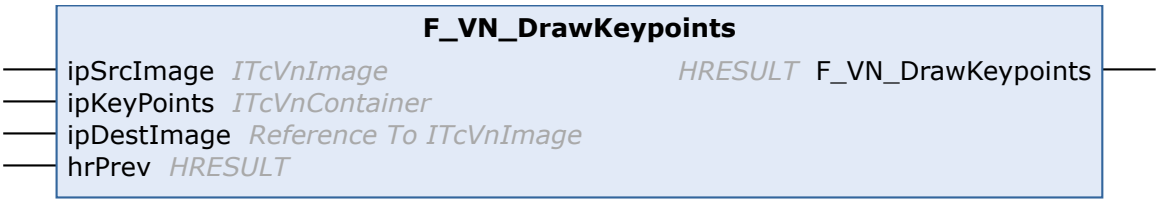
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.17 F_VN_DrawKeypoints



Draws the keypoints for visualization purpose.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_DrawKeypoints : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipKeyPoints : ITcVnContainer;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (elements of type USINT)
ipKeyPoints	ITcVnContainer [▶ 385]	Container with the keypoints (ContainerType_Vector_TcVnKeyPoint)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate color image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

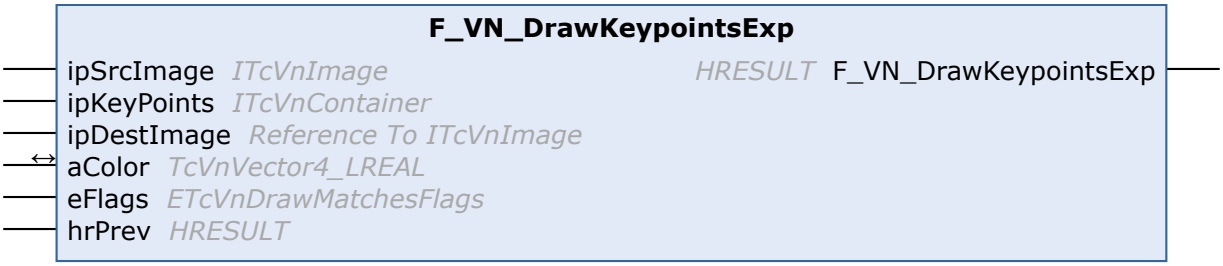
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.18 F_VN_DrawKeypointsExp



Draws the keypoints for visualization purpose. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_DrawKeypointsExp : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipKeyPoints : ITcVnContainer;
    ipDestImage : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    eFlags      : ETcVnDrawMatchesFlags;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (elements of type USINT)
ipKeyPoints	ITcVnContainer [▶ 385]	Container with the keypoints (ContainerType_Vector_TcVnKeyPoint)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate color image will be created if required.)
eFlags	ETcVnDrawMatchesFlags [▶ 207]	A combination of flags to support overdrawing an existing destination image and/or drawing additional (Rich-)Keypoint information (size and orientation)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color to draw the keypoints (for {-1, -1, -1, -1}, a random color is chosen for each point)

Return value

HRESULT [▶ 150]

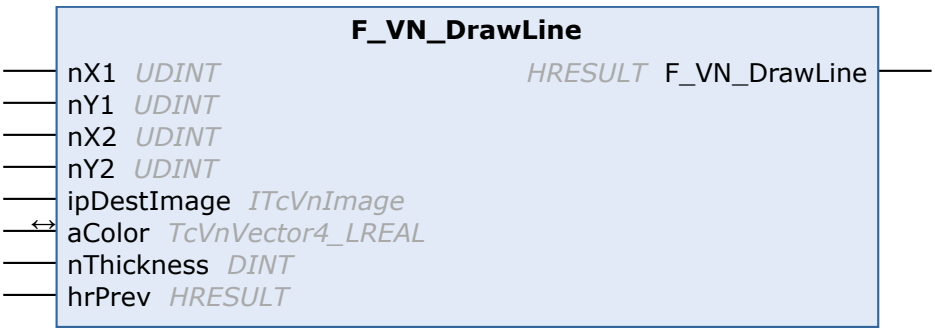
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.19 F_VN_DrawLine



Draws a line.

Syntax

Definition:

```
FUNCTION F_VN_DrawLine : HRESULT
VAR_INPUT
    nX1          : UDINT;
    nY1          : UDINT;
    nX2          : UDINT;
    nY2          : UDINT;
```

```
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness  : DINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nX1	UDINT	x coordinate of the start point
nY1	UDINT	y coordinate of the start point
nX2	UDINT	x coordinate of the end point
nY2	UDINT	y coordinate of the end point
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

HRESULT

16#734	OUTOFRANGE	Ein Teil der Linie liegt außerhalb des Bildbereichs. Verwenden Sie die Funktion F_VN_ClipLineToBoundary_ITcVnImage [▶ 1562], um sicherzustellen, dass die angegebene Linie in den Bildbereich passt.

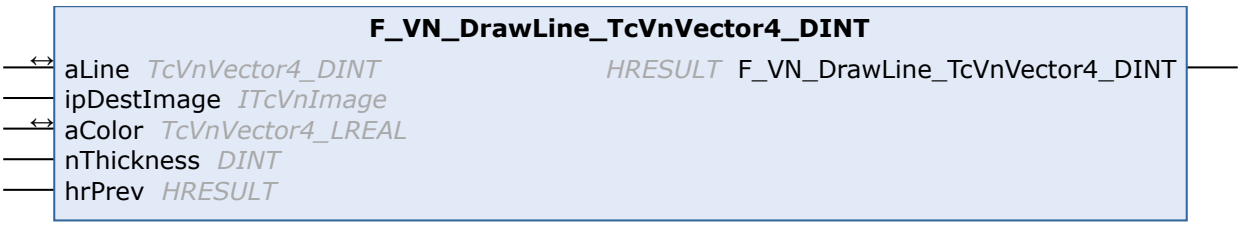
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.20 F_VN_DrawLine_TcVnVector4_DINT



Draws a line.

Syntax

Definition:

```
FUNCTION F_VN_DrawLine_TcVnVector4_DINT : HRESULT
VAR_IN_OUT
    aLine       : TcVnVector4_DINT;
END_VAR
VAR_INPUT
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness  : DINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLine	TcVnVector4_DINT [▶ 173]	The start and end point of a line segment [x1, y1, x2, y2]
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

HRESULT

16#734	OUTOFRANGE	Ein Teil der Linie liegt außerhalb des Bildbereichs. Verwenden Sie die Funktion F_VN_ClipLineToBoundary_ITcVnImage [▶ 1562], um sicherzustellen, dass die angegebene Linie in den Bildbereich passt.

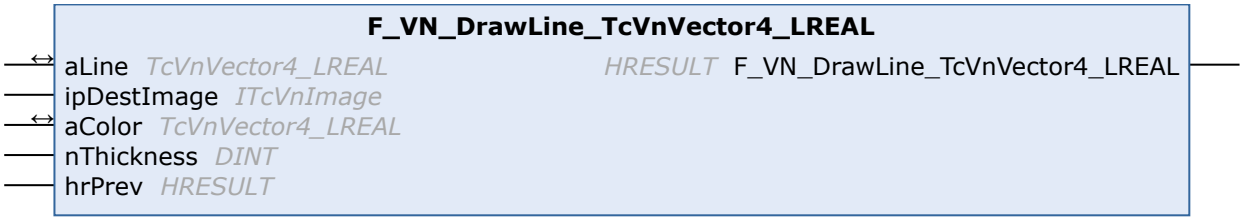
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.21 F_VN_DrawLine_TcVnVector4_LREAL



Draws a line.

Syntax

Definition:

```
FUNCTION F_VN_DrawLine_TcVnVector4_LREAL : HRESULT
VAR_IN_OUT
    aLine       : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness  : DINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLine	TcVnVector4_LREAL [▶ 173]	The first and second element describe the x and y component of the direction vector. The third and fourth element describe the x and y component of the position vector.
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

HINWEIS

Floating Point Exceptions

Diese Funktion kann Fehler herbeiführen, wenn die Option **Floating point exceptions** der ausführenden SPS-Task [▶ 63] aktiv ist. Deaktivieren Sie diese Option deshalb.

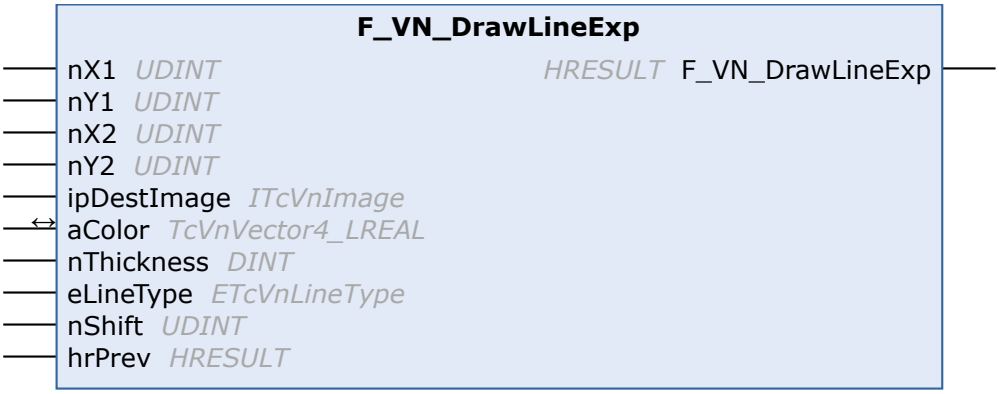
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.22 F_VN_DrawLineExp



Draws a line. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_DrawLineExp : HRESULT
VAR_INPUT
    nX1          : UDINT;
    nY1          : UDINT;
    nX2          : UDINT;
    nY2          : UDINT;
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor       : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness   : DINT;
    eLineType    : ETcVnLineType;
    nShift       : UDINT;
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nX1	UDINT	x coordinate of the start point
nY1	UDINT	y coordinate of the start point
nX2	UDINT	x coordinate of the end point
nY2	UDINT	y coordinate of the end point
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness
eLineType	ETcVnLineType [▶ 219]	Line type
nShift	UDINT	Fractional bits of the coordinates (bit shift)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

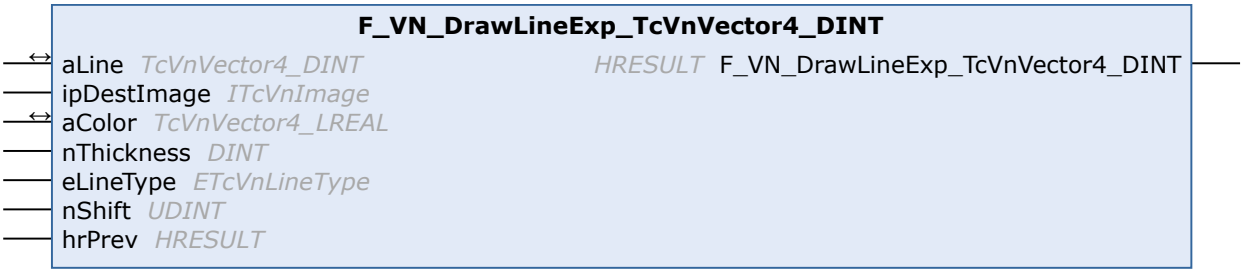
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.23 F_VN_DrawLineExp_TcVnVector4_DINT



Draws a line. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_DrawLineExp_TcVnVector4_DINT : HRESULT
VAR_IN_OUT
    aLine       : TcVnVector4_DINT;
END_VAR
VAR_INPUT
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness  : DINT;
    eLineType   : ETcVnLineType;
    nShift      : UDINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness
eLineType	ETcVnLineType [▶ 219]	Line type
nShift	UDINT	Fractional bits of the coordinates (bit shift)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLine	TcVnVector4_DINT [▶ 173]	The start and end point of a line segment [x1, y1, x2, y2]
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

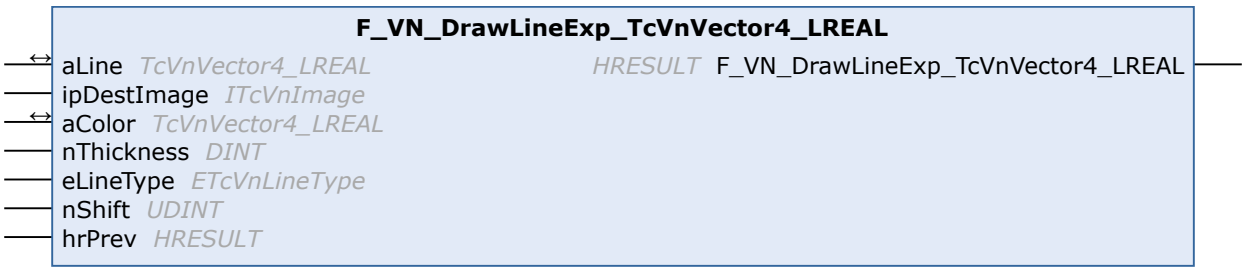
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.24 F_VN_DrawLineExp_TcVnVector4_LREAL



Draws a line. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_DrawLineExp_TcVnVector4_LREAL : HRESULT
VAR_IN_OUT
    aLine       : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness  : DINT;
```

```
    eLineType   : ETcVnLineType;
    nShift      : UDINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness
eLineType	ETcVnLineType [▶ 219]	Line type
nShift	UDINT	Fractional bits of the coordinates (bit shift)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLine	TcVnVector4_LREAL [▶ 173]	The first and second element describe the x and y component of the direction vector. The third and fourth element describe the x and y component of the position vector.
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

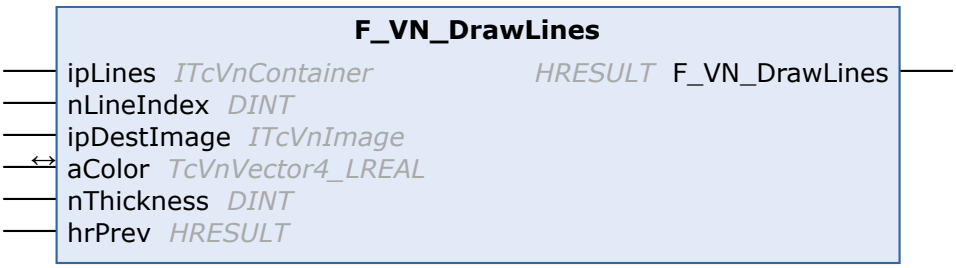
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.25 F_VN_DrawLines



Draws lines.

Syntax

Definition:

```
FUNCTION F_VN_DrawLines : HRESULT
VAR_INPUT
    ipLines     : ITcVnContainer;
    nLineIndex  : DINT;
```

```
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness  : DINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipLines	ITcVnContainer [▶ 385]	Container with line descriptions (ContainerType_Vector_TcVnVector2_REAL: the distance from the origin [0] in pixels and the rotation angle [1] in radians. ContainerType_Vector_TcVnVector4_LREAL: the first and second element describe the x and y component of a vector collinear to the line. The third and fourth element describe the x and y component of a point on the line. ContainerType_Vector_TcVnVector4_DINT: start and end point [x1, y1, x2, y2])
nLineIndex	DINT	Index of a specific line to be drawn (if negative, all lines within the container are drawn)
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

HRESULT

16#734	OUTOFRANGE	Ein Teil der Linie liegt außerhalb des Bildbereichs. Verwenden Sie die Funktion F_VN_ClipLineToBoundary_ITcVnImage [▶ 1562], um sicherzustellen, dass die angegebene Linie in den Bildbereich passt.

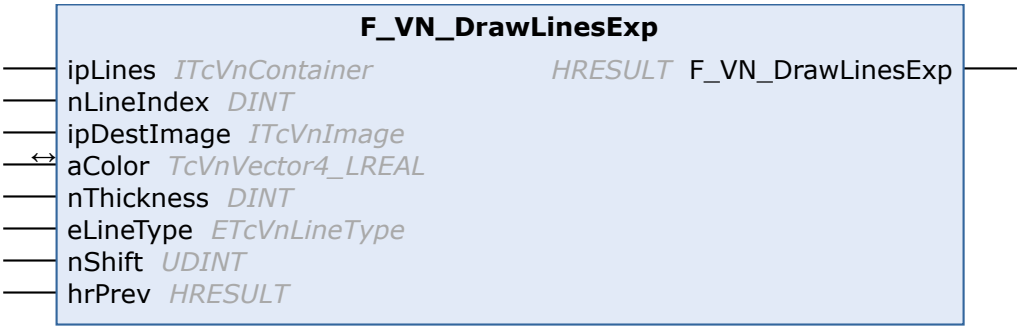
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.26 F_VN_DrawLinesExp



Draws lines. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_DrawLinesExp : HRESULT
VAR_INPUT
    ipLines     : ITcVnContainer;
    nLineIndex  : DINT;
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness  : DINT;
    eLineType   : ETcVnLineType;
    nShift      : UDINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipLines	ITcVnContainer [▶ 385]	Container with line descriptions (ContainerType_Vector_TcVnVector2_REAL: the distance from the origin [0] in pixels and the rotation angle [1] in radians. ContainerType_Vector_TcVnVector4_LREAL: the first and second element describe the x and y component of a vector collinear to the line. The third and fourth element describe the x and y component of a point on the line. ContainerType_Vector_TcVnVector4_DINT: start and end point [x1, y1, x2, y2])
nLineIndex	DINT	Index of a specific line to be drawn (if negative, all lines within the container are drawn)
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness
eLineType	ETcVnLineType [▶ 219]	Line type
nShift	UDINT	Fractional bits of the coordinates (bit shift)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

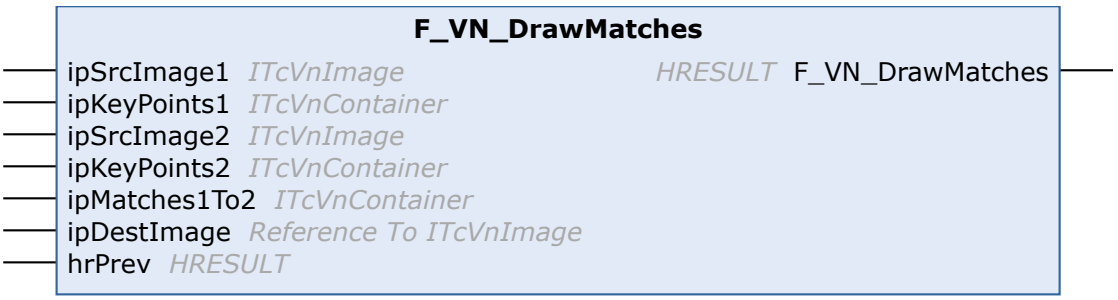
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.27 F_VN_DrawMatches



Draws the keypoints and matches for visualization purpose.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_DrawMatches : HRESULT
VAR_INPUT
    ipSrcImage1   : ITcVnImage;
    ipKeyPoints1  : ITcVnContainer;
    ipSrcImage2   : ITcVnImage;
    ipKeyPoints2  : ITcVnContainer;
    ipMatches1To2 : ITcVnContainer;
    ipDestImage   : Reference To ITcVnImage;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage1	ITcVnImage [▶ 426]	Source image 1 (elements of type USINT)
ipKeyPoints1	ITcVnContainer [▶ 385]	Container with the keypoints 1 (ContainerType_Vector_TcVnKeyPoint)
ipSrcImage2	ITcVnImage [▶ 426]	Source image 2 (elements of type USINT, must have the same number of channels as ipSrcImage1)
ipKeyPoints2	ITcVnContainer [▶ 385]	Container with the keypoints 2 (ContainerType_Vector_TcVnKeyPoint)
ipMatches1To2	ITcVnContainer [▶ 385]	Container with the descriptor matches (ContainerType_Vector_TcVnDMatch)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate color image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

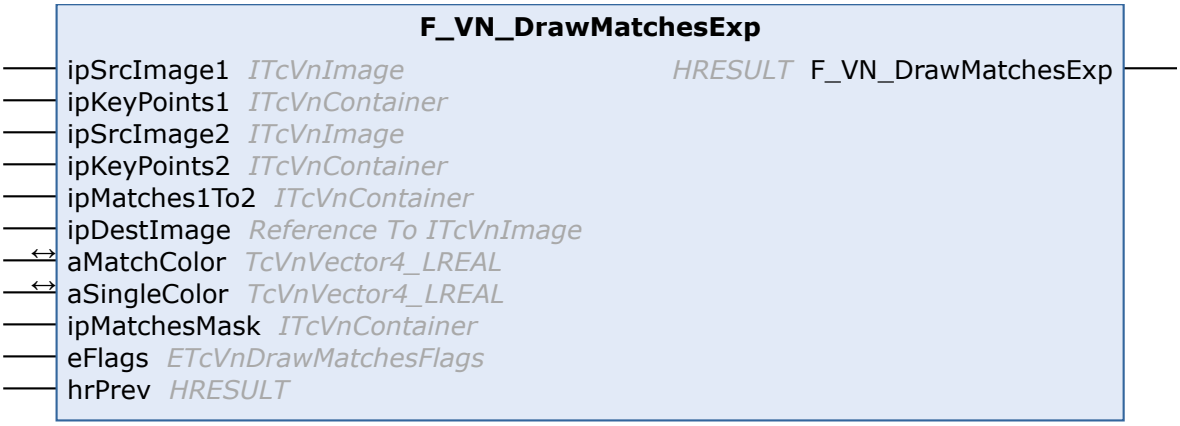
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.28 F_VN_DrawMatchesExp



Draws the keypoints and matches for visualization purpose. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_DrawMatchesExp : HRESULT
VAR_INPUT
    ipSrcImage1   : ITcVnImage;
    ipKeyPoints1  : ITcVnContainer;
    ipSrcImage2   : ITcVnImage;
    ipKeyPoints2  : ITcVnContainer;
    ipMatches1To2 : ITcVnContainer;
    ipDestImage   : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    aMatchColor   : TcVnVector4_LREAL;
    aSingleColor  : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipMatchesMask : ITcVnContainer;
    eFlags        : ETcVnDrawMatchesFlags;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage1	ITcVnImage [▶ 426]	Source image 1 (elements of type USINT)
ipKeyPoints1	ITcVnContainer [▶ 385]	Container with the keypoints 1 (ContainerType_Vector_TcVnKeyPoint)
ipSrcImage2	ITcVnImage [▶ 426]	Source image 2 (elements of type USINT, must have the same number of channels as ipSrcImage1)
ipKeyPoints2	ITcVnContainer [▶ 385]	Container with the keypoints 2 (ContainerType_Vector_TcVnKeyPoint)
ipMatches1To2	ITcVnContainer [▶ 385]	Container with the descriptor matches (ContainerType_Vector_TcVnDMatch)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate color image will be created if required.)
ipMatchesMask	ITcVnContainer [▶ 385]	Mask to select the matches to be drawn (ContainerType_Vector_SINT; Set to 0 if all matches should be drawn.)
eFlags	ETcVnDrawMatchesFlags [▶ 207]	A combination of flags to support overdrawing an existing destination image and/or drawing additional (Rich-)Keypoint information (size and orientation) and/or skipping single keypoints
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMatchColor	TcVnVector4_LREAL [▶ 173]	Color to draw the matches (for {-1, -1, -1, -1}, a random color is chosen for each point and line)
aSingleColor	TcVnVector4_LREAL [▶ 173]	Color to draw the single keypoints (for {-1, -1, -1, -1}, a random color is chosen for each point)

Return value

HRESULT [▶ 150]

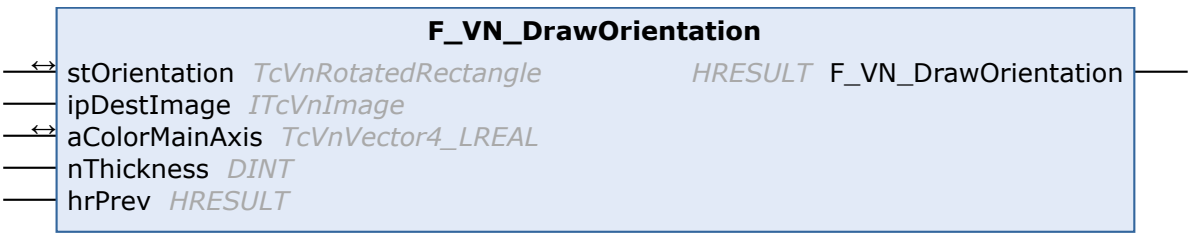
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.29 F_VN_DrawOrientation



Draw the main axis of a set of points/contour based on a rotated rectangle.

Syntax

Definition:

```
FUNCTION F_VN_DrawOrientation : HRESULT
VAR_IN_OUT
    stOrientation  : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    ipDestImage    : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColorMainAxis : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness     : DINT;
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stOrientation	TcVnRotatedRectangle [▶ 258]	Rotated rectangle, containig center of mass, lenghts of axes, and rotation angle in clockwise direction
aColorMainAxis	TcVnVector4_LREAL [▶ 173]	Color of the major axis

Return value

HRESULT [▶ 150]

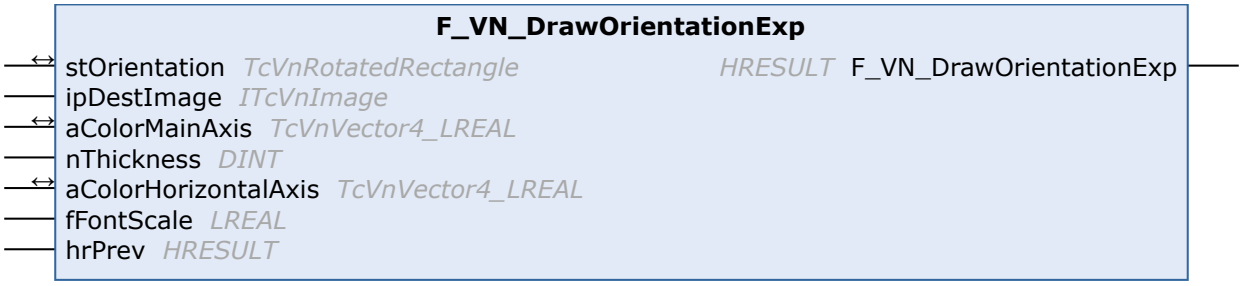
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.30 F_VN_DrawOrientationExp



Draw the main axis, the horizontal axis and the angle of a set of points/contour based on a rotated rectangle. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_DrawOrientationExp : HRESULT
VAR_IN_OUT
    stOrientation        : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    ipDestImage          : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColorMainAxis       : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness           : DINT;
END_VAR
VAR_IN_OUT
    aColorHorizontalAxis : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    fFontScale           : LREAL;
    hrPrev               : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness
fFontScale	LREAL	Scaling factor for the text
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stOrientation	TcVnRotatedRectangle [▶ 258]	Rotated rectangle, containig center of mass, lenghts of axes, and rotation angle in clockwise direction
aColorMainAxis	TcVnVector4_LREAL [▶ 173]	Color of the major axis
aColorHorizontalAxis	TcVnVector4_LREAL [▶ 173]	Color of the horizontal axis

Return value

HRESULT [▶ 150]

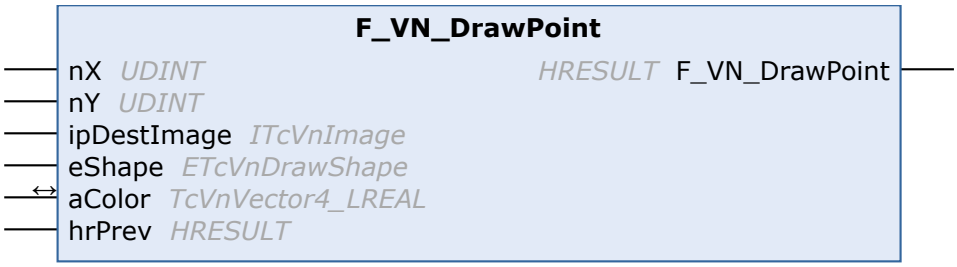
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.31 F_VN_DrawPoint



Draws a point.

Syntax

Definition:

```
FUNCTION F_VN_DrawPoint : HRESULT
VAR_INPUT
    nX          : UDINT;
    nY          : UDINT;
    ipDestImage : ITcVnImage;
    eShape      : ETcVnDrawShape;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nX	UDINT	x coordinate of the point
nY	UDINT	y coordinate of the point
ipDestImage	ITcVnImage [▶ 426]	Destination image
eShape	ETcVnDrawShape [▶ 208]	Point shape
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color (for {-1, -1, -1, -1}, a random color is chosen)

Return value

HRESULT [▶ 150]

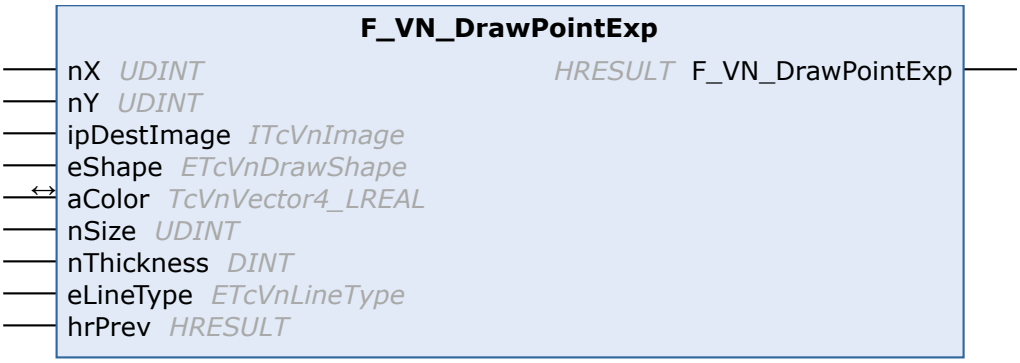
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.32 F_VN_DrawPointExp



Draw a point. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_DrawPointExp : HRESULT
VAR_INPUT
    nX          : UDINT;
    nY          : UDINT;
    ipDestImage : ITcVnImage;
    eShape      : ETcVnDrawShape;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nSize       : UDINT;
    nThickness  : DINT;
    eLineType   : ETcVnLineType;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nX	UDINT	x coordinate of the point
nY	UDINT	y coordinate of the point
ipDestImage	ITcVnImage [▶ 426]	Destination image
eShape	ETcVnDrawShape [▶ 208]	Point shape
nSize	UDINT	Size of the shape (half width)
nThickness	DINT	Line thickness (if negative, the shape is filled if it is closed)
eLineType	ETcVnLineType [▶ 219]	Line type
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color (for {-1, -1, -1, -1}, a random color is chosen)

Return value

HRESULT [▶ 150]

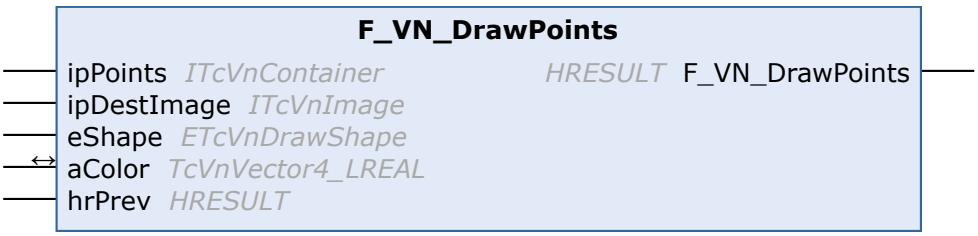
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.33 F_VN_DrawPoints



Draw a collection of points.

Syntax

Definition:

```
FUNCTION F_VN_DrawPoints : HRESULT
VAR_INPUT
    ipPoints     : ITcVnContainer;
    ipDestImage : ITcVnImage;
    eShape       : ETcVnDrawShape;
END_VAR
VAR_IN_OUT
    aColor       : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipPoints	ITcVnContainer [▶ 385]	Container with TcVnPoint2_DINT or TcVnPoint2_REAL elements
ipDestImage	ITcVnImage [▶ 426]	Destination image
eShape	ETcVnDrawShape [▶ 208]	Point shape
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color (for {-1, -1, -1, -1}, a random color is chosen for each point)

Return value

HRESULT [▶ 150]

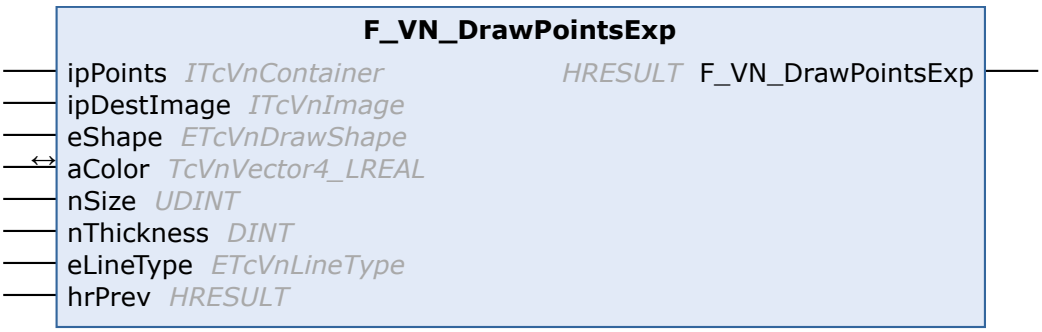
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.34 F_VN_DrawPointsExp



Draw a collection of points. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_DrawPointsExp : HRESULT
VAR_INPUT
    ipPoints    : ITcVnContainer;
    ipDestImage : ITcVnImage;
    eShape      : ETcVnDrawShape;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nSize       : UDINT;
    nThickness  : DINT;
    eLineType   : ETcVnLineType;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipPoints	ITcVnContainer [▶ 385]	Container with TcVnPoint2_DINT or TcVnPoint2_REAL elements
ipDestImage	ITcVnImage [▶ 426]	Destination image
eShape	ETcVnDrawShape [▶ 208]	Point shape
nSize	UDINT	Size of the shape (half width)
nThickness	DINT	Line thickness (if negative, the shape is filled if it is closed)
eLineType	ETcVnLineType [▶ 219]	Line type
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color (for {-1, -1, -1, -1}, a random color is chosen for each point)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.35 F_VN_DrawPolygon

Draw a pologon using collection of points.

Syntax

Definition:

```
FUNCTION F_VN_DrawPolygon : HRESULT
VAR_INPUT
    ipPoints    : ITcVnContainer;
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
```

```
    nThickness  : DINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipPoints	ITcVnContainer [▶ 385]	Container with TcVnPoint2_DINT, TcVnPoint2_REAL, or TcVnPoint2_LREAL elements
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness (if negative, the polygon is filled)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

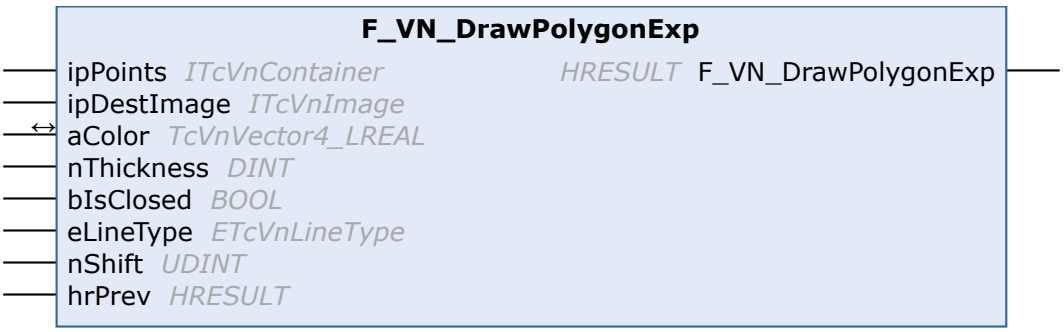
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.36 F_VN_DrawPolygonExp



Draw a pologon using collection of points. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_DrawPolygonExp : HRESULT
VAR_INPUT
    ipPoints    : ITcVnContainer;
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
```

```
END_VAR
VAR_INPUT
    nThickness   : DINT;
    bIsClosed    : BOOL;
    eLineType    : ETcVnLineType;
    nShift       : UDINT;
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipPoints	ITcVnContainer [▶ 385]	Container with TcVnPoint2_DINT, TcVnPoint2_REAL, or TcVnPoint2_LREAL elements
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness (if negative, the shape is filled)
bIsClosed	BOOL	If it is true, the function draws a line from the last point to the first point to close the polygon.
eLineType	ETcVnLineType [▶ 219]	Line type
nShift	UDINT	Fractional bits of the coordinates (bit shift)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

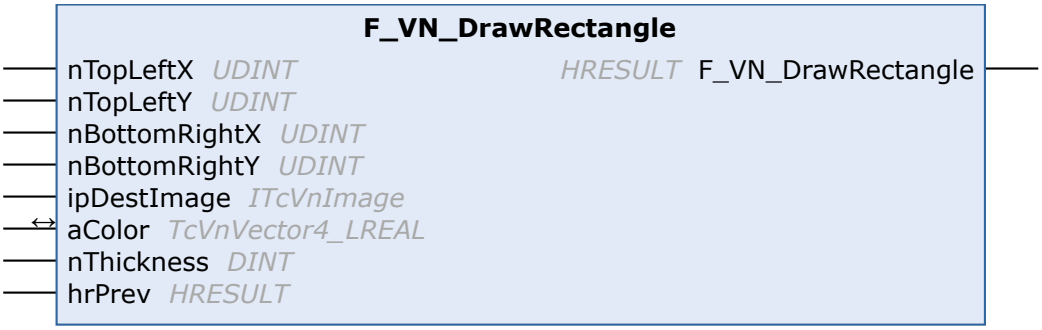
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.37 F_VN_DrawRectangle



Draw a rectangle.

Syntax

Definition:

```
FUNCTION F_VN_DrawRectangle : HRESULT
VAR_INPUT
    nTopLeftX     : UDINT;
    nTopLeftY     : UDINT;
    nBottomRightX : UDINT;
    nBottomRightY : UDINT;
    ipDestImage   : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor        : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness    : DINT;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nTopLeftX	UDINT	x coordinate of the top left corner
nTopLeftY	UDINT	y coordinate of the top left corner
nBottomRightX	UDINT	x coordinate of the bottom right corner
nBottomRightY	UDINT	y coordinate of the bottom right corner
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness (if negative, the rectangle is filled)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

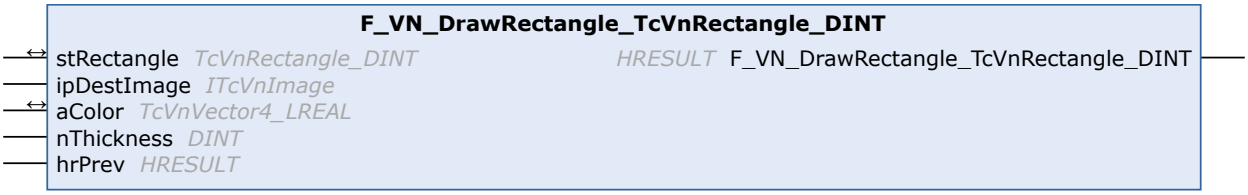
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.38 F_VN_DrawRectangle_TcVnRectangle_DINT



Draw a rectangle.

Syntax

Definition:

```
FUNCTION F_VN_DrawRectangle_TcVnRectangle_DINT : HRESULT
VAR_IN_OUT
    stRectangle : TcVnRectangle_DINT;
END_VAR
VAR_INPUT
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness  : DINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness (if negative, the rectangle is filled)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRectangle	TcVnRectangle_DINT [▶ 257]	Rectangle to be drawn
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.39 F_VN_DrawRectangle_TcVnRectangle_UDINT

Draw a rectangle.

Syntax

Definition:

```
FUNCTION F_VN_DrawRectangle_TcVnRectangle_UDINT : HRESULT
VAR_IN_OUT
    stRectangle : TcVnRectangle_UDINT;
END_VAR
VAR_INPUT
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness  : DINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness (if negative, the rectangle is filled)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRectangle	TcVnRectangle_UDINT [▶ 258]	Rectangle to be drawn
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

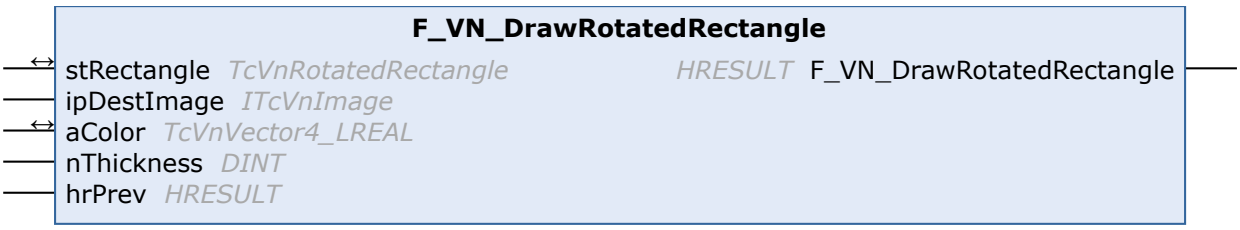
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.40 F_VN_DrawRotatedRectangle



Draw a rectangle.

Syntax

Definition:

```
FUNCTION F_VN_DrawRotatedRectangle : HRESULT
VAR_IN_OUT
    stRectangle : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness  : DINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness (if negative, the rectangle is filled)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRectangle	TcVnRotatedRectangle [▶ 258]	Rectangle to be drawn (rotation angle in degrees)
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

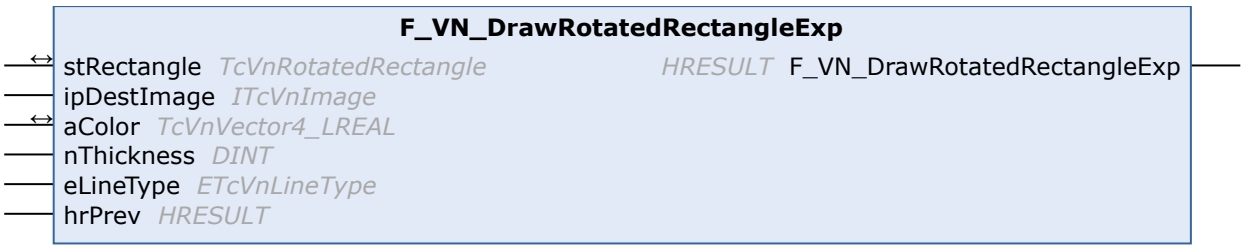
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.41 F_VN_DrawRotatedRectangleExp



Draw a rectangle. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_DrawRotatedRectangleExp : HRESULT
VAR_IN_OUT
    stRectangle : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness  : DINT;
    eLineType   : ETcVnLineType;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipDestImage	ITcVnImage [▶ 426]	Destination image
nThickness	DINT	Line thickness (if negative, the rectangle is filled)
eLineType	ETcVnLineType [▶ 219]	Line type
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRectangle	TcVnRotatedRectangle [▶ 258]	Rectangle to be drawn (rotation angle in degrees)
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

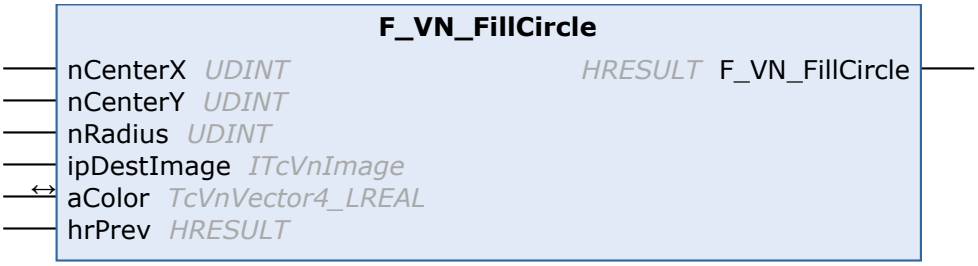
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.42 F_VN_FillCircle



Paint a filled circle.

Syntax

Definition:

```
FUNCTION F_VN_FillCircle : HRESULT
VAR_INPUT
    nCenterX    : UDINT;
    nCenterY    : UDINT;
    nRadius     : UDINT;
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nCenterX	UDINT	x coordinate of the center
nCenterY	UDINT	y coordinate of the center
nRadius	UDINT	Radius
ipDestImage	ITcVnImage [▶ 426]	Destination image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.43 F_VN_FillContours

Paint a single point set or multiple point sets that are interpreted as contours.

Syntax

Definition:

```
FUNCTION F_VN_FillContours : HRESULT
VAR_INPUT
    ipContours    : ITcVnContainer;
    nContourIndex : DINT;
    ipDestImage   : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor        : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipContours	ITcVnContainer [▶ 385]	Single contour (ContainerType_Vector_TcVnPoint2_DINT) or multiple contours (ContainerType_Vector_Vector_TcVnPoint2_DINT)
nContourIndex	DINT	Index of a specific contour to be drawn (if negative, all contours within the container are drawn)
ipDestImage	ITcVnImage [▶ 426]	Destination image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.44 F_VN_FillEllipse

Paint a filled ellipse.

Syntax

Definition:

```
FUNCTION F_VN_FillEllipse : HRESULT
VAR_IN_OUT
    stEllipse   : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipDestImage	ITcVnImage [▶ 426]	Destination image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stEllipse	TcVnRotatedRectangle [▶ 258]	Ellipse to be painted (rotation angle in degrees)
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

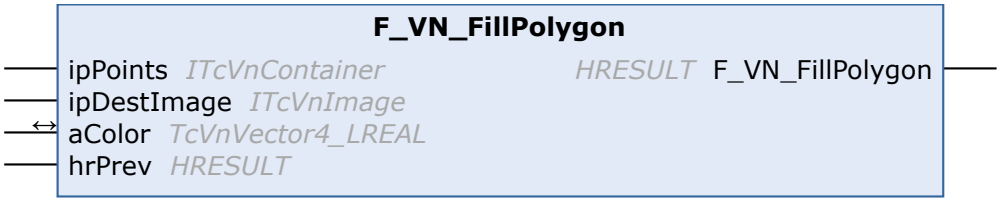
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.45 F_VN_FillPolygon



Paint a filled pologon using a set of points.

Syntax

Definition:

```
FUNCTION F_VN_FillPolygon : HRESULT
VAR_INPUT
    ipPoints    : ITcVnContainer;
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipPoints	ITcVnContainer [▶ 385]	Container with TcVnPoint2_DINT, TcVnPoint2_REAL, or TcVnPoint2_LREAL elements
ipDestImage	ITcVnImage [▶ 426]	Destination image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

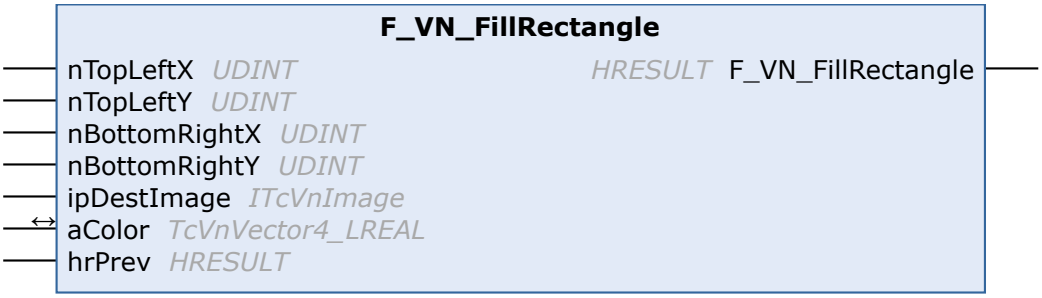
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.46 F_VN_FillRectangle



Paint a filled rectangle.

Syntax

Definition:

```
FUNCTION F_VN_FillRectangle : HRESULT
VAR_INPUT
    nTopLeftX     : UDINT;
    nTopLeftY     : UDINT;
    nBottomRightX : UDINT;
    nBottomRightY : UDINT;
    ipDestImage   : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor        : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nTopLeftX	UDINT	x coordinate of the top left corner
nTopLeftY	UDINT	y coordinate of the top left corner
nBottomRightX	UDINT	x coordinate of the bottom right corner
nBottomRightY	UDINT	y coordinate of the bottom right corner
ipDestImage	ITcVnImage [▶ 426]	Destination image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.47 F_VN_FillRotatedRectangle

Paint a filled rectangle.

Syntax

Definition:

```
FUNCTION F_VN_FillRotatedRectangle : HRESULT
VAR_IN_OUT
    stRectangle : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    ipDestImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipDestImage	ITcVnImage [▶ 426]	Destination image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRectangle	TcVnRotatedRectangle [▶ 258]	Rectangle to be painted (rotation angle in degrees)
aColor	TcVnVector4_LREAL [▶ 173]	Color

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.48 F_VN_PlotIntensityProfile

Plots the pixel intensity profile along a line segment in an image.

Syntax

Definition:

```
FUNCTION F_VN_PlotIntensityProfile : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    aStartPoint : TcVnPoint2_REAL;
    aEndPoint   : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (USINT, 1 channel. x: position, y: intensity, origin: bottom left. An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aStartPoint	TcVnPoint2_REAL [▶ 171]	Start point of the line segment
aEndPoint	TcVnPoint2_REAL [▶ 171]	End point of the line segment

Return value

HRESULT [▶ 150]

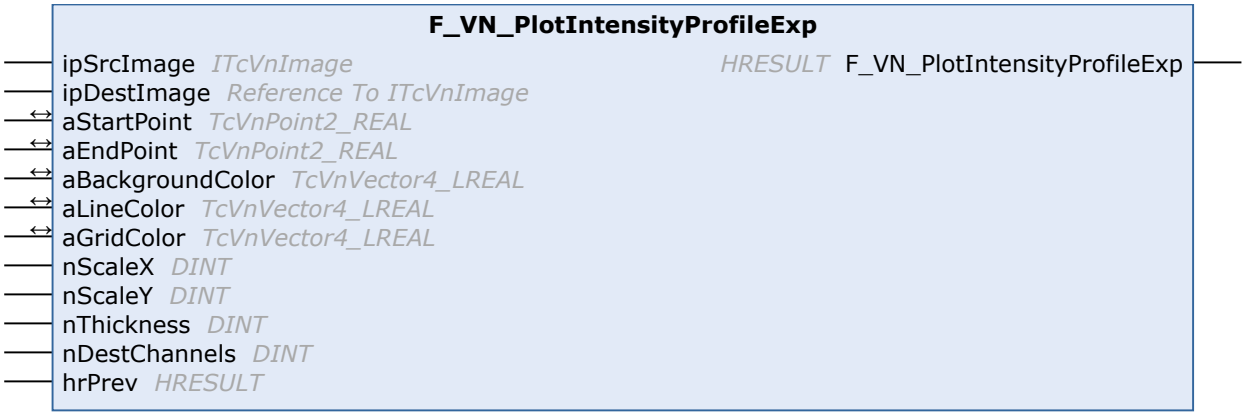
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.49 F_VN_PlotIntensityProfileExp



Plots the pixel intensity profile along a line segment in an image. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_PlotIntensityProfileExp : HRESULT
VAR_INPUT
    ipSrcImage       : ITcVnImage;
    ipDestImage      : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    aStartPoint      : TcVnPoint2_REAL;
    aEndPoint        : TcVnPoint2_REAL;
    aBackgroundColor : TcVnVector4_LREAL;
    aLineColor       : TcVnVector4_LREAL;
    aGridColor       : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nScaleX          : DINT;
    nScaleY          : DINT;
    nThickness       : DINT;
    nDestChannels    : DINT;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (USINT, 1 channel. x: position, y: intensity, origin: bottom left. An appropriate destination image will be created if required.)
nScaleX	DINT	Scale in x direction (ipDestImage will have a width of nScaleX * (ipSrcImage width - 1) + 1)
nScaleY	DINT	Scale in y direction (ipDestImage will have a height of nScaleY * 255 + 1)
nThickness	DINT	Line thickness
nDestChannels	DINT	ipDestImage channels (1 or 3)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aStartPoint	TcVnPoint2_REAL [▶ 171]	Start point of the line segment
aEndPoint	TcVnPoint2_REAL [▶ 171]	End point of the line segment
aBackgroundColor	TcVnVector4_LREAL [▶ 173]	Background color
aLineColor	TcVnVector4_LREAL [▶ 173]	Line color
aGridColor	TcVnVector4_LREAL [▶ 173]	Grid color

Return value

HRESULT [▶ 150]

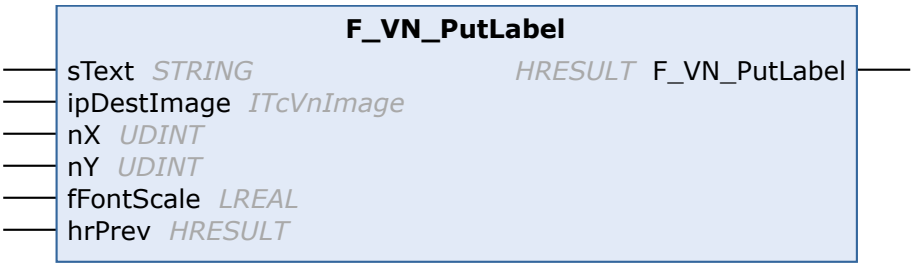
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.50 F_VN_PutLabel



Write a label (black text on white background) into an image.

Syntax

Definition:

```
FUNCTION F_VN_PutLabel : HRESULT
VAR_INPUT
    sText       : STRING;
    ipDestImage : ITcVnImage;
    nX          : UDINT;
    nY          : UDINT;
    fFontScale  : LREAL;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
sText	STRING	Text
ipDestImage	ITcVnImage [▶ 426]	Destination image
nX	UDINT	x coordinate (bottom left)
nY	UDINT	y coordinate (bottom left)
fFontScale	LREAL	Scaling factor
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

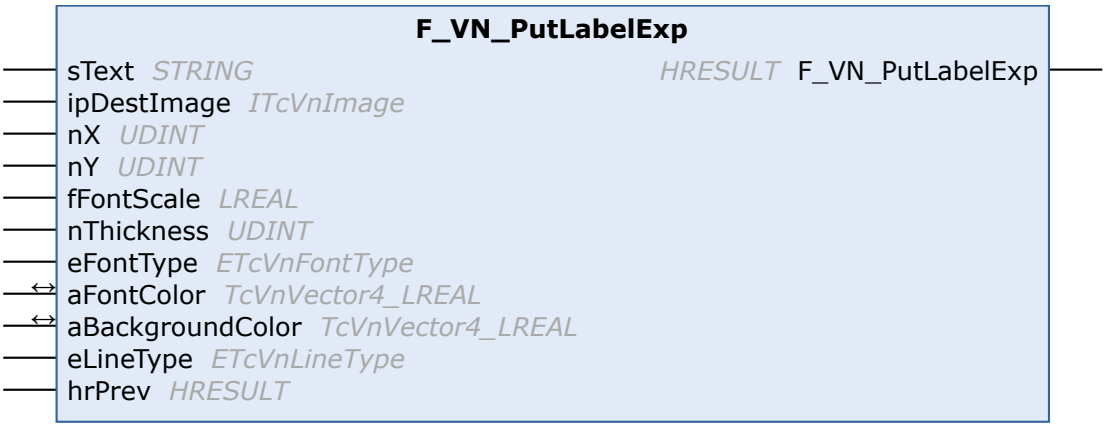
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.51 F_VN_PutLabelExp



Write a label (text on unified background) into an image. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_PutLabelExp : HRESULT
VAR_INPUT
    sText            : STRING;
    ipDestImage      : ITcVnImage;
    nX               : UDINT;
    nY               : UDINT;
    fFontScale       : LREAL;
    nThickness       : UDINT;
    eFontType        : ETcVnFontType;
END_VAR
VAR_IN_OUT
    aFontColor       : TcVnVector4_LREAL;
    aBackgroundColor : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
```

```
    eLineType         : ETcVnLineType;
    hrPrev            : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
sText	STRING	Text
ipDestImage	ITcVnImage [▶ 426]	Destination image
nX	UDINT	x coordinate (bottom left)
nY	UDINT	y coordinate (bottom left)
fFontScale	LREAL	Scaling factor
nThickness	UDINT	Line thickness
eFontType	ETcVnFontType [▶ 215]	Font type
eLineType	ETcVnLineType [▶ 219]	Line type
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aFontColor	TcVnVector4_LREAL [▶ 173]	Font color
aBackgroundColor	TcVnVector4_LREAL [▶ 173]	Background color

Return value

HRESULT [▶ 150]

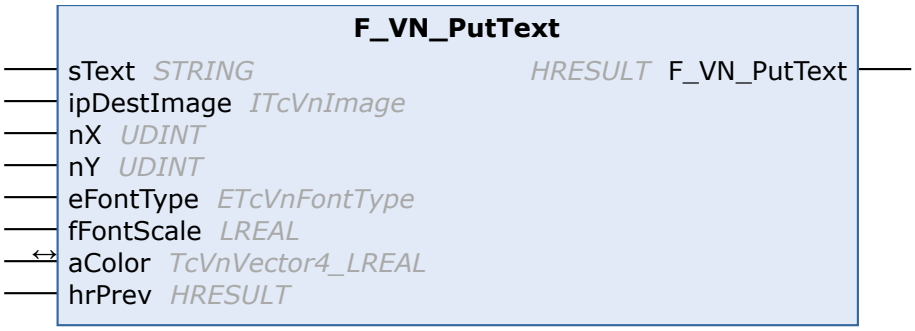
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.52 F_VN_PutText



Write text into an image.

Syntax

Definition:

```
FUNCTION F_VN_PutText : HRESULT
VAR_INPUT
    sText       : STRING;
    ipDestImage : ITcVnImage;
    nX          : UDINT;
    nY          : UDINT;
    eFontType   : ETcVnFontType;
    fFontScale  : LREAL;
END_VAR
VAR_IN_OUT
    aColor      : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
sText	STRING	Text
ipDestImage	ITcVnImage [▶ 426]	Destination image
nX	UDINT	x coordinate (bottom left)
nY	UDINT	y coordinate (bottom left)
eFontType	ETcVnFontType [▶ 215]	Font type
fFontScale	LREAL	Scaling factor
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Text color

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Folgende Schriftarten stehen zur Verfügung:

Verwandte Funktionen

- F_VN_PutLabelExp [▶ 1083]
- F_VN_PutText(Exp) [▶ 1084]

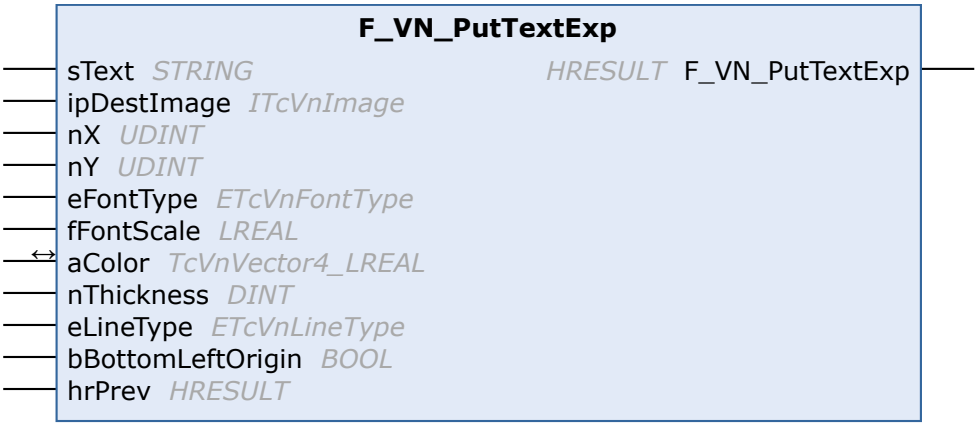
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.11.53 F_VN_PutTextExp



Write text into an image. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_PutTextExp : HRESULT
VAR_INPUT
    sText             : STRING;
    ipDestImage       : ITcVnImage;
    nX                : UDINT;
    nY                : UDINT;
    eFontType         : ETcVnFontType;
    fFontScale        : LREAL;
END_VAR
VAR_IN_OUT
    aColor            : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nThickness        : DINT;
    eLineType         : ETcVnLineType;
    bBottomLeftOrigin : BOOL;
    hrPrev            : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
sText	STRING	Text
ipDestImage	ITcVnImage [▶ 426]	Destination image
nX	UDINT	x coordinate (bottom left)
nY	UDINT	y coordinate (bottom left)
eFontType	ETcVnFontType [▶ 215]	Font type
fFontScale	LREAL	Scaling factor
nThickness	DINT	Line thickness
eLineType	ETcVnLineType [▶ 219]	Line type
bBottomLeftOrigin	BOOL	Sets the image origin to the bottom left corner, if true
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aColor	TcVnVector4_LREAL [▶ 173]	Text color

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.12 Fourier Analysis

Diese Gruppe enthält Funktionen zur Frequenzanalyse und Frequenzfilterung in Bildern.

Funktionen

Butterworth-Filter

- F_VN_CreateBandpassButterworthFilter [▶ 1088]
- F_VN_CreateBandrejectButterworthFilter [▶ 1090]
- F_VN_CreateHighpassButterworthFilter [▶ 1093]
- F_VN_CreateLowpassButterworthFilter [▶ 1095]

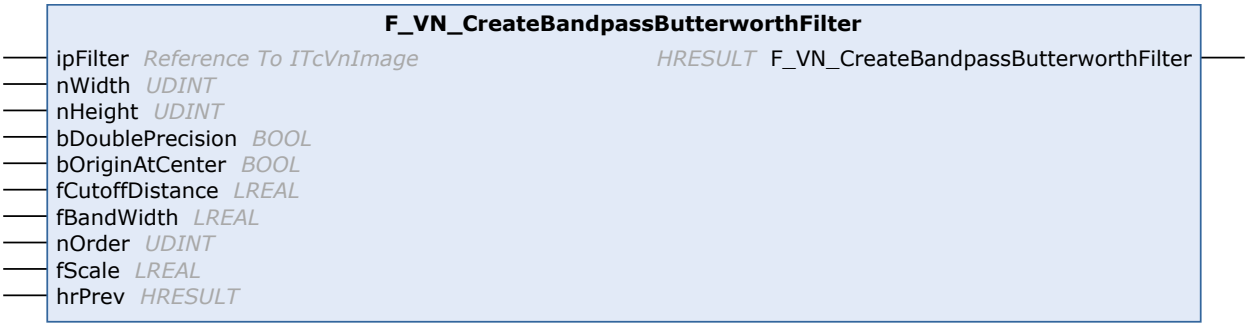
Gauß-Filter

- F_VN_CreateBandpassGaussianFilter [▶ 1089]
- F_VN_CreateBandrejectGaussianFilter [▶ 1091]
- F_VN_CreateHighpassGaussianFilter [▶ 1094]
- F_VN_CreateLowpassGaussianFilter [▶ 1096]

Diskrete Fourier-Transformation

- F_VN_Dft [▶ 1097]
- F_VN_InverseDft [▶ 1098]
- F_VN_OptimalDftSize [▶ 1099]

6.1.4.12.1 F_VN_CreateBandpassButterworthFilter



Creates a bandpass Butterworth filter, which can be applied to a frequency domain image by element-wise multiplication.

Syntax

Definition:

```
FUNCTION F_VN_CreateBandpassButterworthFilter : HRESULT
VAR_INPUT
    ipFilter         : Reference To ITcVnImage;
    nWidth           : UDINT;
    nHeight          : UDINT;
    bDoublePrecision : BOOL;
    bOriginAtCenter  : BOOL;
    fCutoffDistance  : LREAL;
    fBandWidth       : LREAL;
    nOrder           : UDINT;
    fScale           : LREAL;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipFilter	Reference To ITcVnImage [▶ 426]	Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).
nWidth	UDINT	Filter width (even, >= 2, must match the width of the image it is applied to)
nHeight	UDINT	Filter height (even, >= 2, must match the height of the image it is applied to)
bDoublePrecision	BOOL	If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).
bOriginAtCenter	BOOL	If true, the filter origin is shifted to the image center.
fCutoffDistance	LREAL	The cutoff distance of the Butterworth filter (> 0).
fBandWidth	LREAL	The band width of the Butterworth filter (> 0).
nOrder	UDINT	The order of the Butterworth filter (> 0).
fScale	LREAL	The scale factor of the Butterworth filter denominator term (> 0, e.g. 0.414 or 1.0).
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

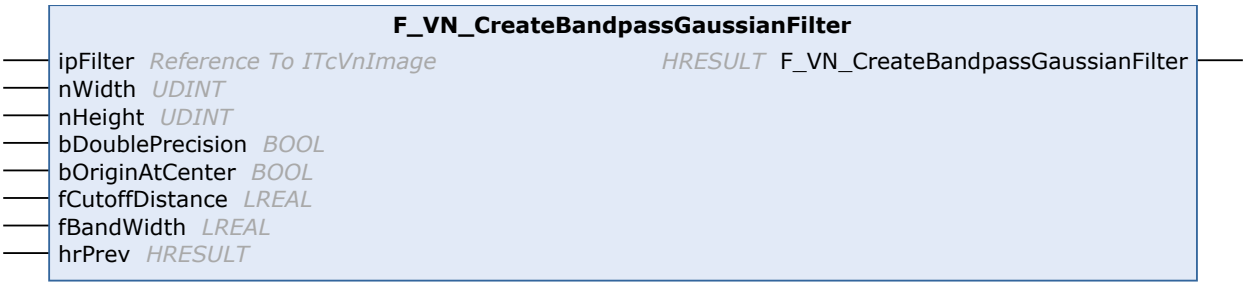
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.12.2 F_VN_CreateBandpassGaussianFilter



Creates a bandpass Gaussian filter, which can be applied to a frequency domain image by element-wise multiplication.

Syntax

Definition:

```
FUNCTION F_VN_CreateBandpassGaussianFilter : HRESULT
VAR_INPUT
    ipFilter         : Reference To ITcVnImage;
    nWidth           : UDINT;
    nHeight          : UDINT;
```

```
    bDoublePrecision : BOOL;
    bOriginAtCenter  : BOOL;
    fCutoffDistance  : LREAL;
    fBandWidth       : LREAL;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipFilter	Reference To ITcVnImage [▶ 426]	Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).
nWidth	UDINT	Filter width (even, >= 2, must match the width of the image it is applied to)
nHeight	UDINT	Filter height (even, >= 2, must match the height of the image it is applied to)
bDoublePrecision	BOOL	If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).
bOriginAtCenter	BOOL	If true, the filter origin is shifted to the image center.
fCutoffDistance	LREAL	The cutoff distance of the Gaussian filter (> 0).
fBandWidth	LREAL	The band width of the Gaussian filter (> 0).
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

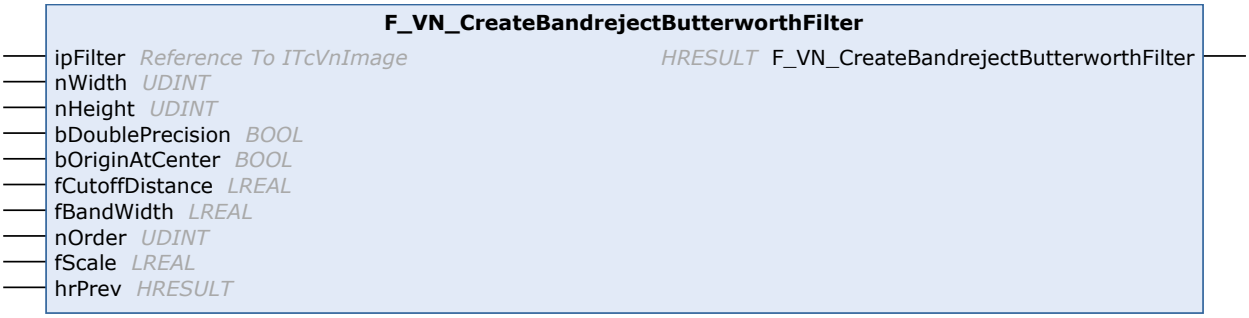
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.12.3 F_VN_CreateBandrejectButterworthFilter



Creates a bandreject Butterworth filter, which can be applied to a frequency domain image by element-wise multiplication.

Syntax

Definition:

```
FUNCTION F_VN_CreateBandrejectButterworthFilter : HRESULT
VAR_INPUT
    ipFilter         : Reference To ITcVnImage;
    nWidth           : UDINT;
    nHeight          : UDINT;
    bDoublePrecision : BOOL;
    bOriginAtCenter  : BOOL;
    fCutoffDistance  : LREAL;
    fBandWidth       : LREAL;
    nOrder           : UDINT;
    fScale           : LREAL;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipFilter	Reference To ITcVnImage [▶ 426]	Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).
nWidth	UDINT	Filter width (even, >= 2, must match the width of the image it is applied to)
nHeight	UDINT	Filter height (even, >= 2, must match the height of the image it is applied to)
bDoublePrecision	BOOL	If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).
bOriginAtCenter	BOOL	If true, the filter origin is shifted to the image center.
fCutoffDistance	LREAL	The cutoff distance of the Butterworth filter (> 0).
fBandWidth	LREAL	The band width of the Butterworth filter (> 0).
nOrder	UDINT	The order of the Butterworth filter (> 0).
fScale	LREAL	The scale factor of the Butterworth filter denominator term (> 0, e.g. 0.414 or 1.0).
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

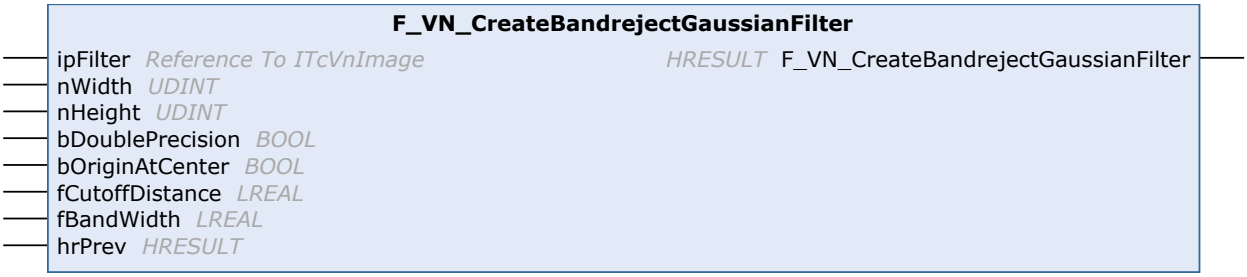
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.12.4 F_VN_CreateBandrejectGaussianFilter



Creates a bandreject Gaussian filter, which can be applied to a frequency domain image by element-wise multiplication.

Syntax

Definition:

```
FUNCTION F_VN_CreateBandrejectGaussianFilter : HRESULT
VAR_INPUT
    ipFilter         : Reference To ITcVnImage;
    nWidth           : UDINT;
    nHeight          : UDINT;
    bDoublePrecision : BOOL;
    bOriginAtCenter  : BOOL;
    fCutoffDistance  : LREAL;
    fBandWidth       : LREAL;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipFilter	Reference To ITcVnImage [▶ 426]	Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).
nWidth	UDINT	Filter width (even, >= 2, must match the width of the image it is applied to)
nHeight	UDINT	Filter height (even, >= 2, must match the height of the image it is applied to)
bDoublePrecision	BOOL	If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).
bOriginAtCenter	BOOL	If true, the filter origin is shifted to the image center.
fCutoffDistance	LREAL	The cutoff distance of the Gaussian filter (> 0).
fBandWidth	LREAL	The band width of the Gaussian filter (> 0).
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

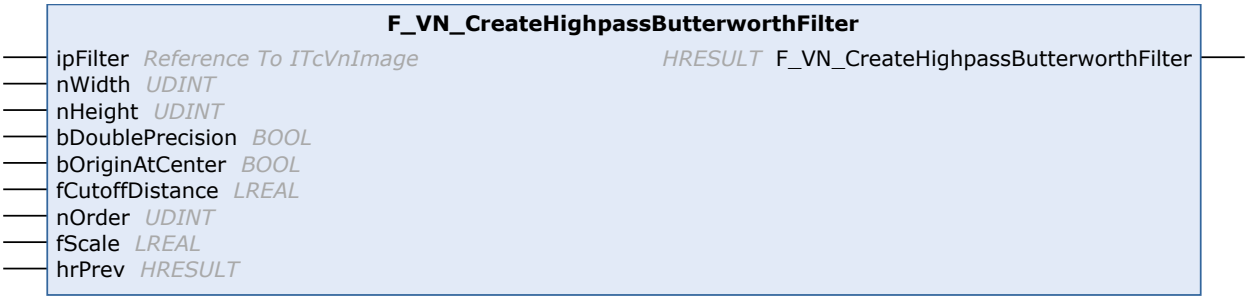
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.12.5 F_VN_CreateHighpassButterworthFilter



Creates a highpass Butterworth filter, which can be applied to a frequency domain image by element-wise multiplication.

Syntax

Definition:

```
FUNCTION F_VN_CreateHighpassButterworthFilter : HRESULT
VAR_INPUT
    ipFilter         : Reference To ITcVnImage;
    nWidth           : UDINT;
    nHeight          : UDINT;
    bDoublePrecision : BOOL;
    bOriginAtCenter  : BOOL;
    fCutoffDistance  : LREAL;
    nOrder           : UDINT;
    fScale           : LREAL;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipFilter	Reference To ITcVnImage [▶ 426]	Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).
nWidth	UDINT	Filter width (even, >= 2, must match the width of the image it is applied to)
nHeight	UDINT	Filter height (even, >= 2, must match the height of the image it is applied to)
bDoublePrecision	BOOL	If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).
bOriginAtCenter	BOOL	If true, the filter origin is shifted to the image center.
fCutoffDistance	LREAL	The cutoff distance of the Butterworth filter (> 0).
nOrder	UDINT	The order of the Butterworth filter (> 0).
fScale	LREAL	The scale factor of the Butterworth filter denominator term (> 0, e.g. 0.414 or 1.0).
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

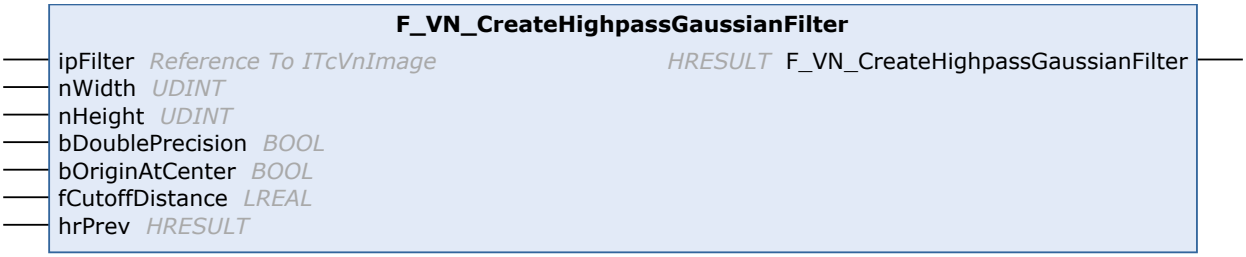
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.12.6 F_VN_CreateHighpassGaussianFilter



Creates a highpass Gaussian filter, which can be applied to a frequency domain image by element-wise multiplication.

Syntax

Definition:

```
FUNCTION F_VN_CreateHighpassGaussianFilter : HRESULT
VAR_INPUT
    ipFilter         : Reference To ITcVnImage;
    nWidth           : UDINT;
    nHeight          : UDINT;
    bDoublePrecision : BOOL;
    bOriginAtCenter  : BOOL;
    fCutoffDistance  : LREAL;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipFilter	Reference To ITcVnImage [▶ 426]	Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).
nWidth	UDINT	Filter width (even, >= 2, must match the width of the image it is applied to)
nHeight	UDINT	Filter height (even, >= 2, must match the height of the image it is applied to)
bDoublePrecision	BOOL	If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).
bOriginAtCenter	BOOL	If true, the filter origin is shifted to the image center.
fCutoffDistance	LREAL	The cutoff distance of the Gaussian filter (> 0).
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

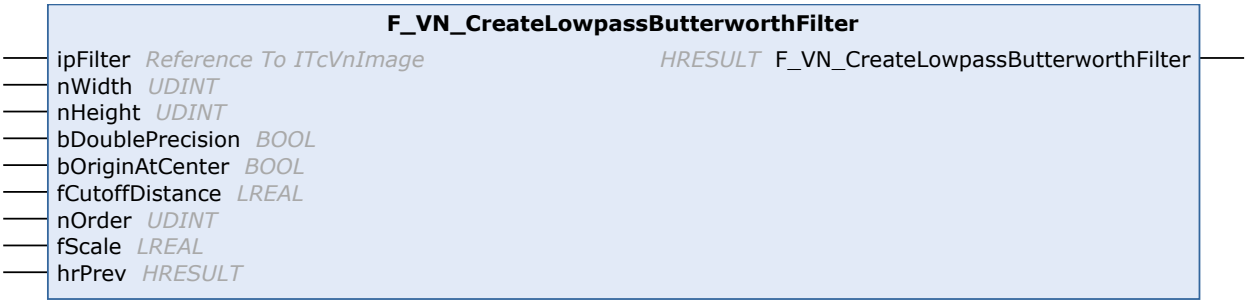
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.12.7 F_VN_CreateLowpassButterworthFilter



Creates a lowpass Butterworth filter, which can be applied to a frequency domain image by element-wise multiplication.

Syntax

Definition:

```
FUNCTION F_VN_CreateLowpassButterworthFilter : HRESULT
VAR_INPUT
    ipFilter         : Reference To ITcVnImage;
    nWidth           : UDINT;
    nHeight          : UDINT;
    bDoublePrecision : BOOL;
    bOriginAtCenter  : BOOL;
    fCutoffDistance  : LREAL;
    nOrder           : UDINT;
    fScale           : LREAL;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipFilter	Reference To ITcVnImage [▶ 426]	Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).
nWidth	UDINT	Filter width (even, >= 2, must match the width of the image it is applied to)
nHeight	UDINT	Filter height (even, >= 2, must match the height of the image it is applied to)
bDoublePrecision	BOOL	If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).
bOriginAtCenter	BOOL	If true, the filter origin is shifted to the image center.
fCutoffDistance	LREAL	The cutoff distance of the Butterworth filter (> 0).
nOrder	UDINT	The order of the Butterworth filter (> 0).
fScale	LREAL	The scale factor of the Butterworth filter denominator term (> 0, e.g. 0.414 or 1.0).
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

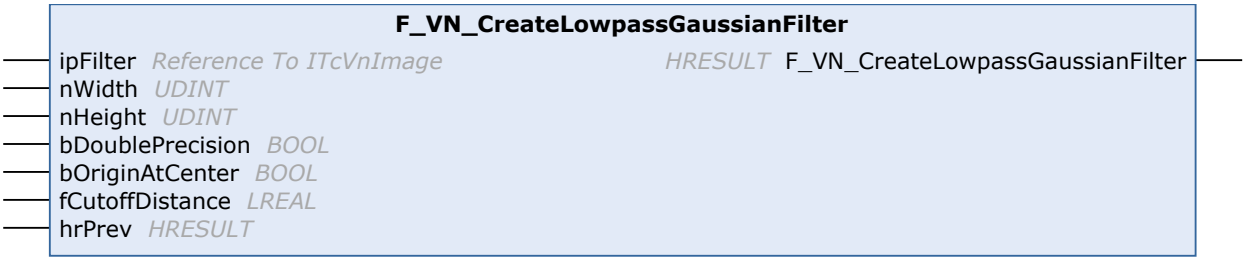
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.12.8 F_VN_CreateLowpassGaussianFilter



Creates a lowpass Gaussian filter, which can be applied to a frequency domain image by element-wise multiplication.

Syntax

Definition:

```
FUNCTION F_VN_CreateLowpassGaussianFilter : HRESULT
VAR_INPUT
    ipFilter         : Reference To ITcVnImage;
    nWidth           : UDINT;
    nHeight          : UDINT;
    bDoublePrecision : BOOL;
    bOriginAtCenter  : BOOL;
    fCutoffDistance  : LREAL;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipFilter	Reference To ITcVnImage [▶ 426]	Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).
nWidth	UDINT	Filter width (even, >= 2, must match the width of the image it is applied to)
nHeight	UDINT	Filter height (even, >= 2, must match the height of the image it is applied to)
bDoublePrecision	BOOL	If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).
bOriginAtCenter	BOOL	If true, the filter origin is shifted to the image center.
fCutoffDistance	LREAL	The cutoff distance of the Gaussian filter (> 0).
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

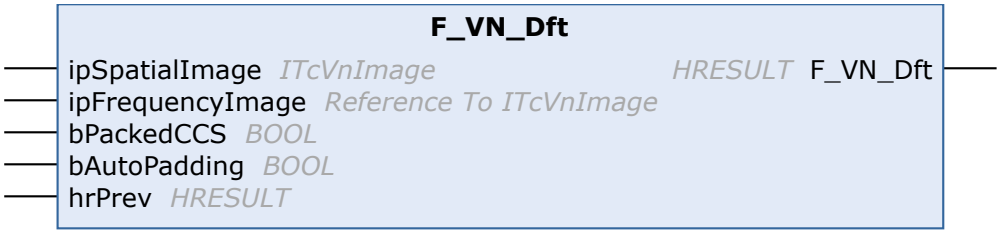
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.12.9 F_VN_Dft



Computes the DFT for a given image.

Syntax

Definition:

```
FUNCTION F_VN_Dft : HRESULT
VAR_INPUT
    ipSpatialImage    : ITcVnImage;
    ipFrequencyImage : Reference To ITcVnImage;
    bPackedCCS        : BOOL;
    bAutoPadding      : BOOL;
    hrPrev            : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSpatialImage	ITcVnImage [▶ 426]	Source image (elements of type REAL or LREAL, 1 (Re) or 2 (Re + Im) channels)
ipFrequencyImage	Reference To ITcVnImage [▶ 426]	Destination image (same type as ipSpatialImage, but number of channels and size can vary depending on bPackedCCS and bAutoPadding.
bPackedCCS	BOOL	If true and ipSpatialImage has only 1 channel, the result image will have 1 channel with packed complex-conjugate-symmetrical format results. Otherwise, the result image will have 2 separate channels (Re + Im), containing the full spectrum.
bAutoPadding	BOOL	If true, the input image is automatically padded (with 0s) to optimal size if required, to speed up dft (Creates a temporary copy so that ipSpatialImage content stays unchanged, which also requires some additional computation power. Therefore, it is recommended to compare execution times with and without padding.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.12.10 F_VN_InverseDft

Compute the inverse DFT for a given frequency image.

Syntax

Definition:

```
FUNCTION F_VN_InverseDft : HRESULT
VAR_INPUT
    ipFrequencyImage : ITcVnImage;
    ipSpatialImage   : Reference To ITcVnImage;
    bRealOutput      : BOOL;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipFrequencyImage	ITcVnImage [▶ 426]	Source frequency image (elements of type REAL or LREAL, 1 (packed CCS) or 2 (Re + Im) channels)
ipSpatialImage	Reference To ITcVnImage [▶ 426]	Destination image (Same type as ipFrequencyImage, but number of channels can vary depending on bRealOutput. An appropriate image will be created if required.)
bRealOutput	BOOL	Only relevant if ipFrequencyImage has 2 channels. If true, the result image will have only 1 channel. Otherwise, the result image will have 2 separate channels (Re + Im). If ipFrequencyImage has only 1 channel, packed CCS format is assumed and the result image will always have only 1 channel.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

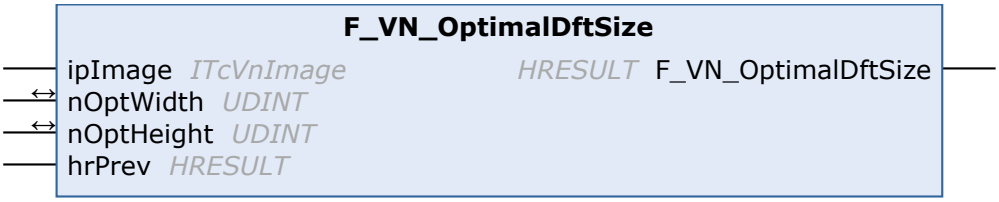
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.12.11 F_VN_OptimalDftSize



Computes the optimal image size for applying a dft (can lead to better performance).

Syntax

Definition:

```
FUNCTION F_VN_OptimalDftSize : HRESULT
VAR_INPUT
    ipImage    : ITcVnImage;
END_VAR
VAR_IN_OUT
    nOptWidth  : UDINT;
    nOptHeight : UDINT;
END_VAR
VAR_INPUT
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Source image for which to compute the optimal width and height
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nOptWidth	UDINT	Returns the optimal width of the image
nOptHeight	UDINT	Returns the optimal height of the image

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13 Geometric and Coordinate Transformations

Diese Gruppe enthält Funktionen zur Transformation von Bildern und Bildpunkten.

Funktionen

Calibration [▶ 1101]

- Intrinsische und extrinsische Kalibrierung
- Kompensation von optischen Verzeichnungen
- Transformation zwischen Bild- und Welt-Koordinaten

Affine Transformation

- F_VN_ApplyRotationToAffineTransformation(Exp) [▶ 1144]
- F_VN_ApplyScalingToAffineTransformation(Exp) [▶ 1146]
- F_VN_ApplyTranslationToAffineTransformation(Exp) [▶ 1148]
- F_VN_ApplyYAxisInversionToAffineTransformation(Exp) [▶ 1150]
- F_VN_DecomposeAffineTransformation(Exp) [▶ 1165]
- F_VN_GenerateAffineTransformationUnitMatrix2D [▶ 1171]
- F_VN_GetAffineTransformation [▶ 1172]
- F_VN_GetAffineTransformation2D(Exp) [▶ 1173]
- F_VN_InvertAffineTransformation [▶ 1178]
- F_VN_WarpAffine(Exp) [▶ 1194]
- F_VN_WarpAffine_Container [▶ 1195]
- F_VN_WarpAffine_Point [▶ 1196]
- F_VN_WarpAffine_Rectangle [▶ 1197]

Transformation zwischen kartesischen und Polar-Koordinaten

- F_VN_ConvertCartesianToPolarAngleImage(Exp) [▶ 1152]
- F_VN_ConvertCartesianToPolarAngles(Exp) [▶ 1153]
- F_VN_ConvertCartesianToPolarImages(Exp) [▶ 1155]
- F_VN_ConvertCartesianToPolarMagnitudeImage [▶ 1157]
- F_VN_ConvertCartesianToPolarMagnitudes [▶ 1158]
- F_VN_ConvertCartesianToPolarPoints(Exp) [▶ 1159]
- F_VN_ConvertPolarToCartesianImages(Exp) [▶ 1161]
- F_VN_ConvertPolarToCartesianPoints(Exp) [▶ 1163]
- F_VN_RemapImageToLogPolarSpace(Exp) [▶ 1182]
- F_VN_RemapImageToPolarSpace(Exp) [▶ 1185]

Einfache Bildoperationen

- F_VN_FlipImage [▶ 1170]
- F_VN_PyramidDown [▶ 1180]
- F_VN_PyramidUp [▶ 1181]
- F_VN_ResizeImage [▶ 1188]

Bildausrichtung und Rotation

- F_VN_AlignRotatedImageRegion(Exp) [▶ 1142]
- F_VN_RotateImage(Exp) [▶ 1191]

Perspektivische Transformation

- F_VN_DecomposeHomography(Exp) [▶ 1167]
- F_VN_GetPerspectiveTransformation [▶ 1175]
- F_VN_Homography(Exp) [▶ 1176]
- F_VN_PerspectiveTransformation [▶ 1179]
- F_VN_WarpPerspective(Exp) [▶ 1199]
- F_VN_WarpPerspective_Container [▶ 1200]
- F_VN_WarpPerspective_Point [▶ 1201]
- F_VN_WarpPerspective_Rectangle [▶ 1202]

6.1.4.13.1 Calibration

Diese Untergruppe enthält Funktionen zur Korrelation zwischen Bildpunkten und Echtweltpunkten. Sie werden üblicherweise zusammen mit den Vermessungsfunktionen [▶ 1511] verwendet.

Funktionen

Kalibration

- F_VN_CalibrateCamera(Exp) [▶ 1102]
- F_VN_CalibrateCameraManually(Exp) [▶ 1112]
- F_VN_CalibrateCameraPlanar(Exp) [▶ 1115]
- F_VN_CalibrateLinescanCamera(Exp) [▶ 1117]
- F_VN_DetectPatternPoints(Exp) [▶ 1127]
- F_VN_GenerateCalibrationPatternReferencePoints [▶ 1130]
- F_VN_SortDetectedPatternPoints [▶ 1135]
- F_VN_SolvePnP(Exp) [▶ 1132]

Kompensation von Linsenverzeichnung

- F_VN_CompensateLensDistortion(Exp) [▶ 1121]
- F_VN_CompensateLensDistortionForPoints(Exp) [▶ 1124]

Transformation zwischen Bild- und Welt-Koordinaten

- F_VN_TransformCoordinatesImageToWorld [▶ 1137]
- F_VN_TransformCoordinatesWorldToImage [▶ 1141]
- F_VN_TransformCoordinatesPlanar [▶ 1139]
- F_VN_ImagePointsWorldDistance [▶ 1131]

Anforderungen an die Eingangsbilder

Die Eingangsbilder sollten nicht in einem gepackten oder gefilterten Format verwendet werden, wie z. B. im Bayer Format. Bei den Funktionen zur Kalibration hat das zur Folge, dass die Erkennung der Muster gestört wird, was zu Schlechteren oder gar keinen Kalibierergebnissen führen kann. Da bei den Kompensationsfunktionen eine Interpolation stattfindet, bei der die ursprüngliche Anordnung der Pixel verändert wird, kann das Bild im Anschluss nicht mehr richtig gewandelt werden. Daher wird empfohlen, eine Konvertierung immer zuerst durchzuführen, bevor eine Funktion aus dieser Untergruppe verwendet wird.

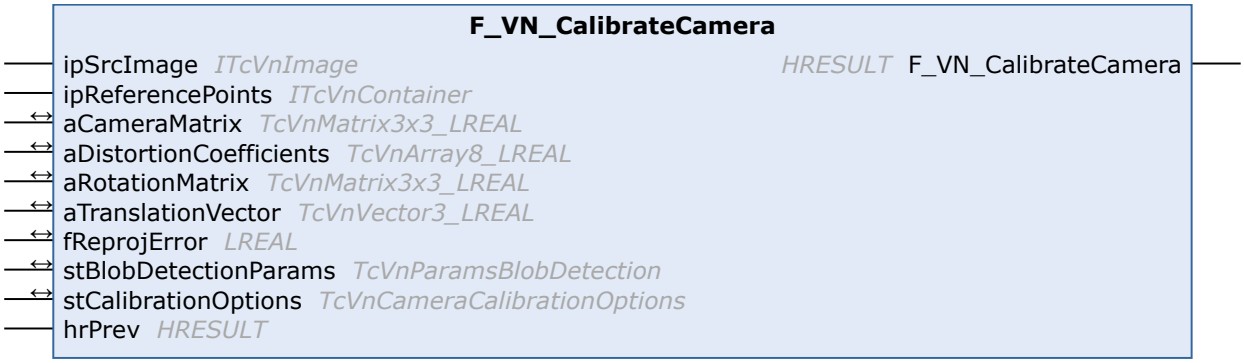
Anforderung an die Kameramatrix

$$CameraMatrix = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Die Kameramatrix setzt sich aus den Brennweiten fx und fy und dem Hauptpunkt (cx,cy) zusammen, die in Pixeleinheiten ausgedrückt werden. Der Hauptpunkt, ist der Schnittpunkt der optischen Achse mit dem Bild und liegt normalerweise in der Nähe der Bildmitte. Folgende Bedingungen müssen eingehalten werden:

- 0 < fx und fy
- cx >= 0 und <= Bildbreite -1
- cy >= 0 und <= Bildhöhe -1
- Die anderen Stellen der Matrix müssen, wie in der Abbildung dargestellt, mit 0 oder 1 gefüllt werden. Andere Werte sind nicht zulässig.

6.1.4.13.1.1 F_VN_CalibrateCamera



Compute the camera parameters (intrinsic + extrinsic) by evaluating an image containing a calibration pattern (circles).
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_CalibrateCamera : HRESULT
VAR_INPUT
    ipSrcImage              : ITcVnImage;
    ipReferencePoints       : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aCameraMatrix           : TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : TcVnArray8_LREAL;
    aRotationMatrix         : TcVnMatrix3x3_LREAL;
    aTranslationVector      : TcVnVector3_LREAL;
    fReprojError            : LREAL;
    stBlobDetectionParams   : TcVnParamsBlobDetection;
    stCalibrationOptions    : TcVnCameraCalibrationOptions;
END_VAR
VAR_INPUT
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image containing a calibration pattern (1 channel, USINT elements)
ipReferencePoints	ITcVnContainer [▶ 385]	Reference calibration pattern point positions (ContainerType_Vector_TcVnPoint3_REAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 173]	Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 173]	Returns the translation vector
fReprojError	LREAL	Returns the reprojection error
stBlobDetectionParams	TcVnParamsBlobDetection [▶ 247]	Parameters for the internally used F_VN_DetectBlobs function
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 240]	Calibration options

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CalibrateCamera` berechnet die intrinsischen und extrinsischen Kameraparameter basierend auf einem Eingangsbild, das ein Kalibriermuster enthält. Dabei müssen die Elemente des Kalibriermusters als Blob detektierbar sein.

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Grauwertbild mit einer Bittiefe von 8 Bit sein. Dieses Bild muss ein Kalibrationsmuster mit Kreisen enthalten, welches mit den Referenz-Punkten `ipReferencePoints` übereinstimmt. Die Funktion erwartet schwarze gefüllte Kreise auf weißem Hintergrund. Zum Invertieren des Bildes kann der `ETcVnThresholdType = TCVN_TT_BINARY_INV` der Blob Funktion verwendet werden.

Für eine Kalibrierung mit mehreren Eingangsbildern wird die Expert-Variante F_VN_CalibrateCameraExp [▶ 1105] benötigt.

Referenz-Punkte

Der Container `ipReferencePoints` muss die Referenz-Punkte enthalten, die auf dem Eingangsbild `ipSrcImage` als Kalibrationsmuster dargestellt sind. Die Punkte müssen als 3-dimensionale Punkte vom Typ `REAL` angegeben werden, sodass der Container den Typ `ContainerType_Vector_TcVnPoint3_REAL` hat.

Die Referenz-Punkte können wie folgt erzeugt werden:

- Manuelles Erstellen einer XML-Datei und Einlesen mit dem Funktionsblock FB_VN_ReadCalibrationPattern [▶ 1642].
- Konfigurieren im Kalibrationsassistenten [▶ 116], Speichern im TcCOM-Objekt durch Klick auf **Write Results** und Einlesen mit der Methode GetCalibPatternRef [▶ 1680] des Funktionsblocks FB_VN_GevCameraControl [▶ 1672]. Der eingestellte Mittelpunkt wird dabei berücksichtigt.
- Referenz-Punkte einzeln in einen Container einfügen.

Matrizen der Kamera-Parameter

Diese Matrizen geben die errechneten Parameter des Kamera-Modells zurück. Sie können daraufhin zum Kompensieren der Objektiv-Verzeichnungen und zum Umrechnen zwischen Pixel- und Weltkoordinaten genutzt werden.

- Kamera-Matrix
- Verzeichnungskoeffizienten
- Rotations-Matrix
- Translations-Vektor

Rundungsfehler

Die errechneten Parameterwerte können aufgrund von Rundungsfehlern geringfügig von den im Kalibrationsassistenten errechneten Werten abweichen.

Reprojektions-Fehler

Parameter für Blob-Detektion

Keine Schachbrettmuster

Da die Referenzpunkte als eine Blob-Struktur detektiert werden, ist die Verwendung von Schachbrettmustern nicht möglich.

Optionen für Kalibrierung

Standardeinstellungen

Um die Standardeinstellungen aus dem Kalibrationsassistenten [▶ 116] zu erhalten, müssen die Bits `bFixAspectRatio`, `bFixPrincipalPoint`, `bFixK5`, `bFixK6` und `bRationalModel` auf TRUE gesetzt werden.

bFixPrincipalPoint = „Fix Center Point“

Das Bit `bFixPrincipalPoint` entspricht der Option „Fix Center Point“ im Kalibrationsassistenten der Kamera.

Beispiele

- Calibration Assistent [▶ 2925]

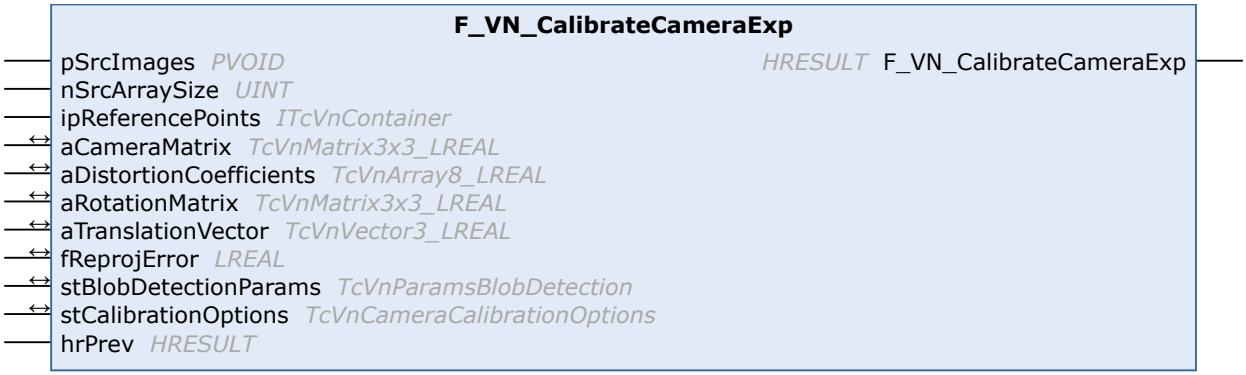
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.2 F_VN_CalibrateCameraExp



Compute the camera parameters (intrinsic + extrinsic) by evaluating images containing a calibration pattern (circles). The extrinsic parameters are computed for the first image in the array. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_CalibrateCameraExp : HRESULT
VAR_INPUT
    pSrcImages              : PVOID;
    nSrcArraySize           : UINT;
    ipReferencePoints       : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aCameraMatrix           : TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : TcVnArray8_LREAL;
    aRotationMatrix         : TcVnMatrix3x3_LREAL;
    aTranslationVector      : TcVnVector3_LREAL;
    fReprojError            : LREAL;
    stBlobDetectionParams   : TcVnParamsBlobDetection;
    stCalibrationOptions    : TcVnCameraCalibrationOptions;
END_VAR
VAR_INPUT
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
pSrcImages	PVOID	Pointer to an array of source images, each containing the same calibration pattern (1 channel, USINT elements)
nSrcArraySize	UINT	pSrcImages array size
ipReferencePoints	ITcVnContainer [▶ 385]	Reference calibration pattern point positions (ContainerType_Vector_TcVnPoint3_REAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 173]	Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 173]	Returns the translation vector
fReprojError	LREAL	Returns the reprojection error
stBlobDetectionParams	TcVnParamsBlobDetection [▶ 247]	Parameters for the internally used F_VN_DetectBlobs function
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 240]	Calibration options

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CalibrateCameraExp` ist die Expert-Variante von F_VN_CalibrateCamera [▶ 1102]. Sie ermöglicht die Verwendung von mehreren Eingangsbildern zur Kalibrierung der Kamera.

Parameter

Eingangsbilder

Die Eingangsbilder für die Kamera-Kalibrierung werden mit Hilfe des Pointers `pSrcImages` und der Anzahl der Bilder `nSrcArraySize` an die Funktion übergeben. Die einzelnen Bilder müssen ansonsten die gleichen Anforderungen erfüllen wie bei der Standard Funktion.

HINWEIS

Pointer-Handhabung

Achten Sie darauf, dass die Angabe des Pointers und der Array-Größe mit den tatsächlichen Daten übereinstimmt. Sonst kann es zu einem unerlaubten Speicherzugriff und einem daraus resultierenden Systemabsturz kommen.

Referenz-Punkte

Der Container `ipReferencePoints` muss die Referenz-Punkte enthalten, die auf dem Eingangsbild `ipSrcImage` als Kalibrationsmuster dargestellt sind. Die Punkte müssen als 3-dimensionale Punkte vom Typ `REAL` angegeben werden, sodass der Container den Typ `ContainerType_Vector_TcVnPoint3_REAL` hat.

Die Referenz-Punkte können wie folgt erzeugt werden:

- Manuelles Erstellen einer XML-Datei und Einlesen mit dem Funktionsblock FB_VN_ReadCalibrationPattern [▶ 1642].
- Konfigurieren im Kalibrationsassistenten [▶ 116], Speichern im TcCOM-Objekt durch Klick auf **Write Results** und Einlesen mit der Methode GetCalibPatternRef [▶ 1680] des Funktionsblocks FB_VN_GevCameraControl [▶ 1672]. Der eingestellte Mittelpunkt wird dabei berücksichtigt.
- Referenz-Punkte einzeln in einen Container einfügen.

Matrizen der Kamera-Parameter

Diese Matrizen geben die errechneten Parameter des Kamera-Modells zurück. Sie können daraufhin zum Kompensieren der Objektiv-Verzeichnungen und zum Umrechnen zwischen Pixel- und Weltkoordinaten genutzt werden.

- Kamera-Matrix
- Verzeichnungskoeffizienten
- Rotations-Matrix
- Translations-Vektor

Rundungsfehler

Die errechneten Parameterwerte können aufgrund von Rundungsfehlern geringfügig von den im Kalibrationsassistenten errechneten Werten abweichen.

Reprojektions-Fehler

Parameter für Blob-Detektion

Keine Schachbrettmuster

Da die Referenzpunkte als eine Blob-Struktur detektiert werden, ist die Verwendung von Schachbrettmustern nicht möglich.

Optionen für Kalibrierung

Standardeinstellungen

Um die Standardeinstellungen aus dem Kalibrationsassistenten [▶ 116] zu erhalten, müssen die Bits `bFixAspectRatio`, `bFixPrincipalPoint`, `bFixK5`, `bFixK6` und `bRationalModel` auf TRUE gesetzt werden.

bFixPrincipalPoint = „Fix Center Point“

Das Bit `bFixPrincipalPoint` entspricht der Option „Fix Center Point“ im Kalibrationsassistenten der Kamera.

Anwendung

Beispiele

- Calibration Assistent [▶ 2925]

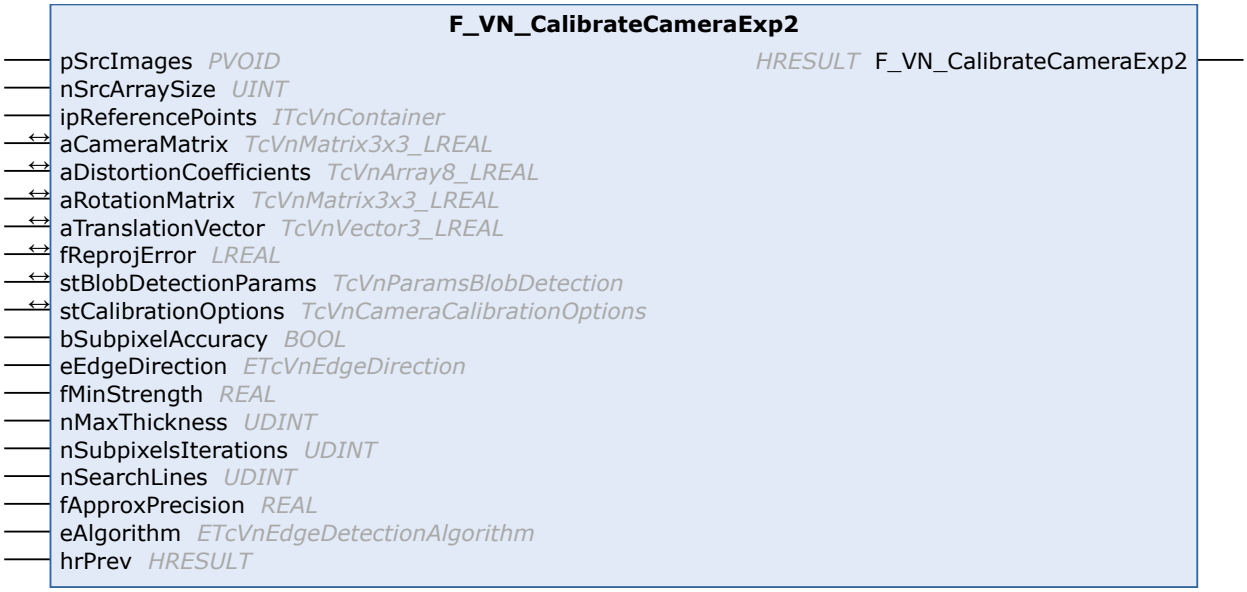
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.3 F_VN_CalibrateCameraExp2



Compute the camera parameters (intrinsic + extrinsic) by evaluating images containing a calibration pattern (circles). The extrinsic parameters are computed for the first image in the array. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_CalibrateCameraExp2 : HRESULT
VAR_INPUT
    pSrcImages              : PVOID;
    nSrcArraySize           : UINT;
    ipReferencePoints       : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aCameraMatrix           : TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : TcVnArray8_LREAL;
    aRotationMatrix         : TcVnMatrix3x3_LREAL;
    aTranslationVector      : TcVnVector3_LREAL;
    fReprojError            : LREAL;
    stBlobDetectionParams   : TcVnParamsBlobDetection;
    stCalibrationOptions    : TcVnCameraCalibrationOptions;
END_VAR
VAR_INPUT
    bSubpixelAccuracy       : BOOL;
    eEdgeDirection          : ETcVnEdgeDirection;
    fMinStrength            : REAL;
    nMaxThickness           : UDINT;
    nSubpixelsIterations    : UDINT;
    nSearchLines            : UDINT;
    fApproxPrecision        : REAL;
    eAlgorithm              : ETcVnEdgeDetectionAlgorithm;
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
pSrcImages	PVOID	Pointer to an array of source images, each containing the same calibration pattern (1 channel, USINT elements)
nSrcArraySize	UINT	pSrcImages array size
ipReferencePoints	ITcVnContainer [▶ 385]	Reference calibration pattern point positions (ContainerType_Vector_TcVnPoint3_REAL)
bSubpixelAccuracy	BOOL	If true, the pattern points are detected with subpixel accuracy
eEdgeDirection	ETcVnEdgeDirection [▶ 210]	Specification of the edge direction to search for (from center to outside ellipse)
fMinStrength	REAL	Specification of the minimum strength (intensity difference) of the edge to search for
nMaxThickness	UDINT	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels
nSubpixelsIterations	UDINT	Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)
nSearchLines	UDINT	Specifies the amount of search lines, which are equally distributed in all directions (must be >= 8 and a multiple of 4)
fApproxPrecision	REAL	Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)
eAlgorithm	ETcVnEdgeDetectionAlgorithm [▶ 209]	Selection of the edge detection algorithm
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 173]	Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 173]	Returns the translation vector
fReprojError	LREAL	Returns the reprojection error
stBlobDetectionParams	TcVnParamsBlobDetection [▶ 247]	Parameters for the internally used F_VN_DetectBlobs function
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 240]	Calibration options

Return value

HRESULT [▶ 150]

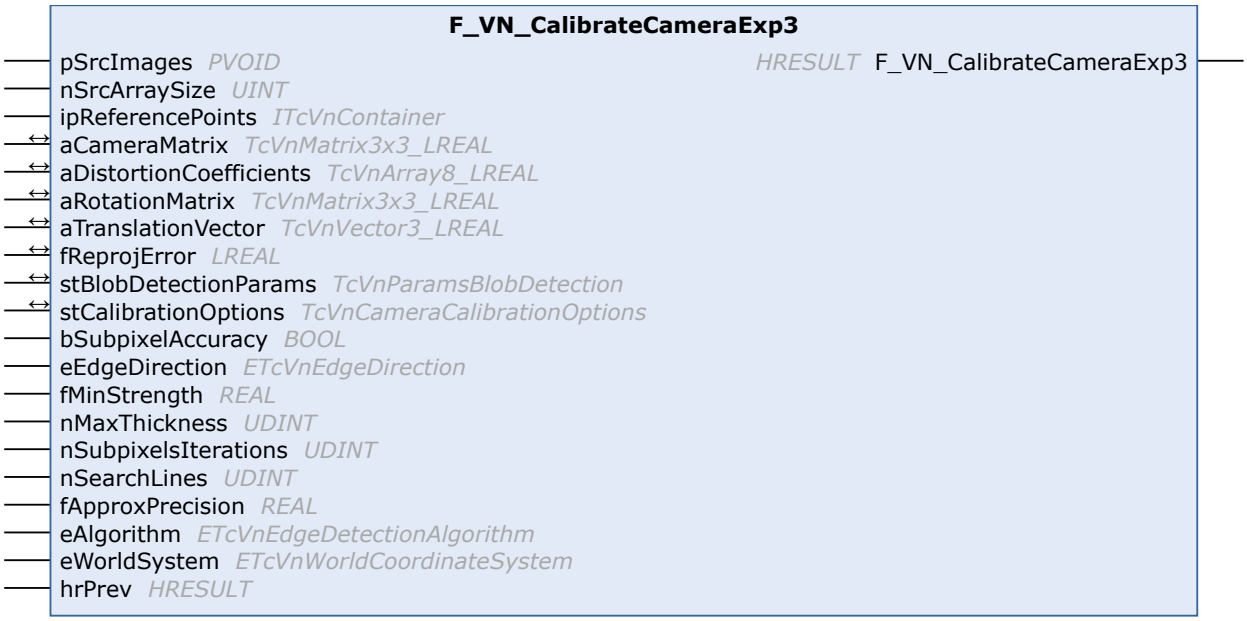
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.4 F_VN_CalibrateCameraExp3



Compute the camera parameters (intrinsic + extrinsic) by evaluating images containing a calibration pattern (circles). The extrinsic parameters are computed for the first image in the array. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_CalibrateCameraExp3 : HRESULT
VAR_INPUT
    pSrcImages              : PVOID;
    nSrcArraySize           : UINT;
    ipReferencePoints       : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aCameraMatrix           : TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : TcVnArray8_LREAL;
    aRotationMatrix         : TcVnMatrix3x3_LREAL;
    aTranslationVector      : TcVnVector3_LREAL;
    fReprojError            : LREAL;
    stBlobDetectionParams   : TcVnParamsBlobDetection;
    stCalibrationOptions    : TcVnCameraCalibrationOptions;
END_VAR
VAR_INPUT
    bSubpixelAccuracy       : BOOL;
    eEdgeDirection          : ETcVnEdgeDirection;
    fMinStrength            : REAL;
    nMaxThickness           : UDINT;
    nSubpixelsIterations    : UDINT;
    nSearchLines            : UDINT;
    fApproxPrecision        : REAL;
    eAlgorithm              : ETcVnEdgeDetectionAlgorithm;
    eWorldSystem            : ETcVnWorldCoordinateSystem;
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
pSrcImages	PVOID	Pointer to an array of source images, each containing the same calibration pattern (1 channel, USINT elements)
nSrcArraySize	UINT	pSrcImages array size
ipReferencePoints	ITcVnContainer [▶ 385]	Reference calibration pattern point positions (ContainerType_Vector_TcVnPoint3_REAL)
bSubpixelAccuracy	BOOL	If true, the pattern points are detected with subpixel accuracy
eEdgeDirection	ETcVnEdgeDirection [▶ 210]	Specification of the edge direction to search for (from center to outside ellipse)
fMinStrength	REAL	Specification of the minimum strength (intensity difference) of the edge to search for
nMaxThickness	UDINT	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels
nSubpixelsIterations	UDINT	Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)
nSearchLines	UDINT	Specifies the amount of search lines, which are equally distributed in all directions (must be >= 8 and a multiple of 4)
fApproxPrecision	REAL	Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)
eAlgorithm	ETcVnEdgeDetectionAlgorithm [▶ 209]	Selection of the edge detection algorithm
eWorldSystem	ETcVnWorldCoordinateSystem [▶ 234]	World coordinate system orientation
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 173]	Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 173]	Returns the translation vector
fReprojError	LREAL	Returns the reprojection error
stBlobDetectionParams	TcVnParamsBlobDetection [▶ 247]	Parameters for the internally used F_VN_DetectBlobs function
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 240]	Calibration options

Return value

HRESULT [▶ 150]

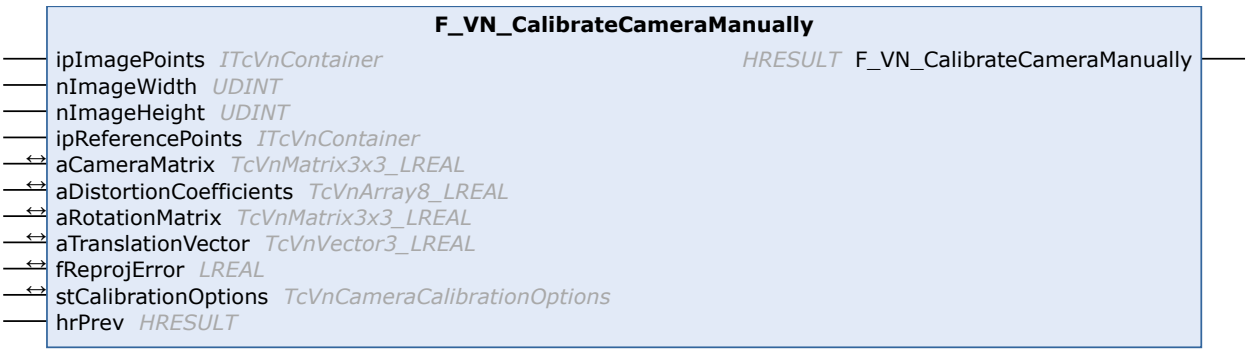
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.5 F_VN_CalibrateCameraManually



Compute the camera parameters (intrinsic + extrinsic). The extrinsic parameters are computed for the first inner container.

Syntax

Definition:

```
FUNCTION F_VN_CalibrateCameraManually : HRESULT
VAR_INPUT
    ipImagePoints           : ITcVnContainer;
    nImageWidth             : UDINT;
    nImageHeight            : UDINT;
    ipReferencePoints       : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aCameraMatrix           : TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : TcVnArray8_LREAL;
    aRotationMatrix         : TcVnMatrix3x3_LREAL;
    aTranslationVector      : TcVnVector3_LREAL;
    fReprojError            : LREAL;
    stCalibrationOptions    : TcVnCameraCalibrationOptions;
END_VAR
VAR_INPUT
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImagePoints	ITcVnContainer [▶ 385]	Provided image points of multiple images (ContainerType_Vector_Vector_TcVnPoint2_REAL, each inner container with at least 6 points). The container and point order must match ipReferencePoints.
nImageWidth	UDINT	Image width
nImageHeight	UDINT	Image height
ipReferencePoints	ITcVnContainer [▶ 385]	Reference world points (ContainerType_Vector_Vector_TcVnPoint3_REAL). The number of inner containers and their amount of points must match ipImagePoints.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 173]	Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns the rotation matrix (for the first inner container)
aTranslationVector	TcVnVector3_LREAL [▶ 173]	Returns the translation vector (for the first inner container)
fReprojError	LREAL	Returns the reprojection error
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 240]	Calibration options

Return value

HRESULT [▶ 150]

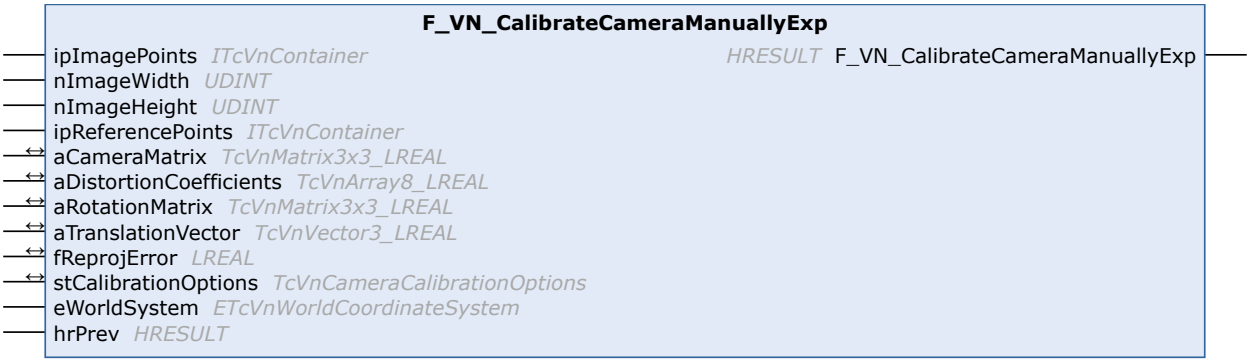
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.6 F_VN_CalibrateCameraManuallyExp



Compute the camera parameters (intrinsic + extrinsic). The extrinsic parameters are computed for the first inner container. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_CalibrateCameraManuallyExp : HRESULT
VAR_INPUT
    ipImagePoints           : ITcVnContainer;
    nImageWidth             : UDINT;
    nImageHeight            : UDINT;
    ipReferencePoints       : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aCameraMatrix           : TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : TcVnArray8_LREAL;
    aRotationMatrix         : TcVnMatrix3x3_LREAL;
    aTranslationVector      : TcVnVector3_LREAL;
    fReprojError            : LREAL;
    stCalibrationOptions    : TcVnCameraCalibrationOptions;
END_VAR
VAR_INPUT
    eWorldSystem            : ETcVnWorldCoordinateSystem;
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImagePoints	ITcVnContainer [▶ 385]	Provided image points of multiple images (ContainerType_Vector_Vector_TcVnPoint2_REAL, each inner container with at least 6 points). The container and point order must match ipReferencePoints.
nImageWidth	UDINT	Image width
nImageHeight	UDINT	Image height
ipReferencePoints	ITcVnContainer [▶ 385]	Reference world points (ContainerType_Vector_Vector_TcVnPoint3_REAL). The number of inner containers and their amount of points must match ipImagePoints.
eWorldSystem	ETcVnWorldCoordinateSystem [▶ 234]	World coordinate system orientation
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 173]	Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns the rotation matrix (for the first inner container)
aTranslationVector	TcVnVector3_LREAL [▶ 173]	Returns the translation vector (for the first inner container)
fReprojError	LREAL	Returns the reprojection error
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 240]	Calibration options

Return value

HRESULT [▶ 150]

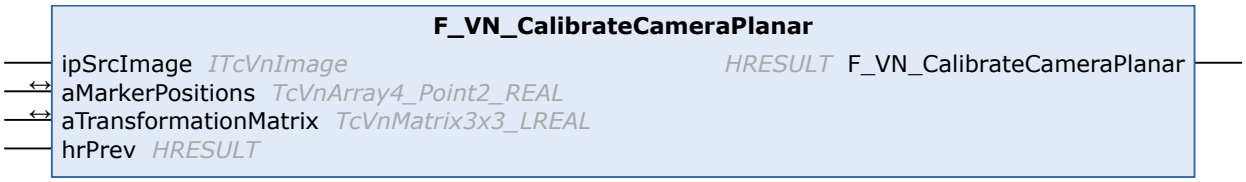
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.7 F_VN_CalibrateCameraPlanar



Calibrate camera using a planar calibration pattern comprised of four circles marking the corners of a rectangle.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_CalibrateCameraPlanar : HRESULT
VAR_INPUT
    ipSrcImage            : ITcVnImage;
END_VAR
VAR_IN_OUT
    aMarkerPositions      : TcVnArray4_Point2_REAL;
    aTransformationMatrix : TcVnMatrix3x3_LREAL;
END_VAR
VAR_INPUT
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Binary source image (background zero, markers non-zero)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMarkerPositions	TcVnArray4_Point2_REAL [▶ 173]	Marker positions within their plane using world units (right-handed coordinate system)
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns a 3-by-3 matrix for the transformation from image coordinates into the reference coordinate system

Return value

HRESULT [▶ 150]

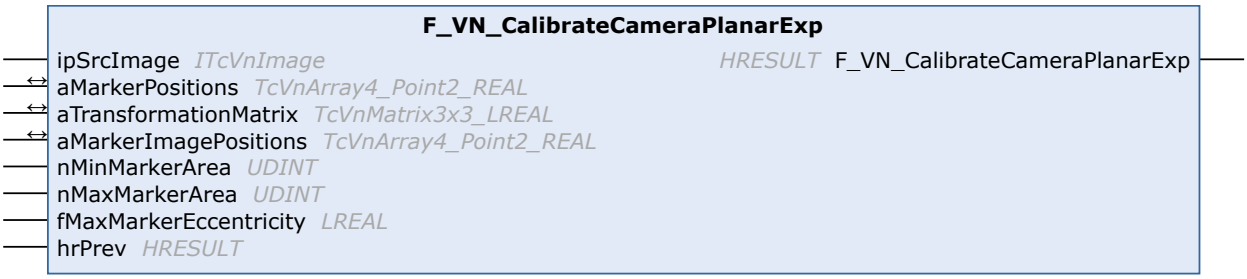
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.8 F_VN_CalibrateCameraPlanarExp



Calibrate camera using a planar calibration pattern comprised of four circles marking the corners of a rectangle. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_CalibrateCameraPlanarExp : HRESULT
VAR_INPUT
    ipSrcImage              : ITcVnImage;
END_VAR
VAR_IN_OUT
    aMarkerPositions        : TcVnArray4_Point2_REAL;
    aTransformationMatrix   : TcVnMatrix3x3_LREAL;
    aMarkerImagePositions   : TcVnArray4_Point2_REAL;
END_VAR
VAR_INPUT
    nMinMarkerArea          : UDINT;
    nMaxMarkerArea          : UDINT;
```

```
    fMaxMarkerEccentricity : LREAL;
    hrPrev                 : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Binary source image (background zero, markers non-zero)
nMinMarkerArea	UDINT	Minimum area for the detection of markers (Smaller markers are ignored.)
nMaxMarkerArea	UDINT	Maximum area for the detection of markers (Larger markers are ignored.)
fMaxMarkerEccentricity	LREAL	Maximum eccenticity for the detection of markers (The eccentricity measures deviations from a round shape (eccentricity = 0: round object; eccentricity = 1.0: linear object).)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMarkerPositions	TcVnArray4_Point2_REAL [▶ 173]	Marker positions within their plane using world units (right-handed coordinate system)
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns a 3-by-3 matrix for the transformation from image coordinates into the reference coordinate system
aMarkerImagePositions	TcVnArray4_Point2_REAL [▶ 173]	Returns the marker positions in image coordinates (left-handed coordinate system)

Return value

HRESULT [▶ 150]

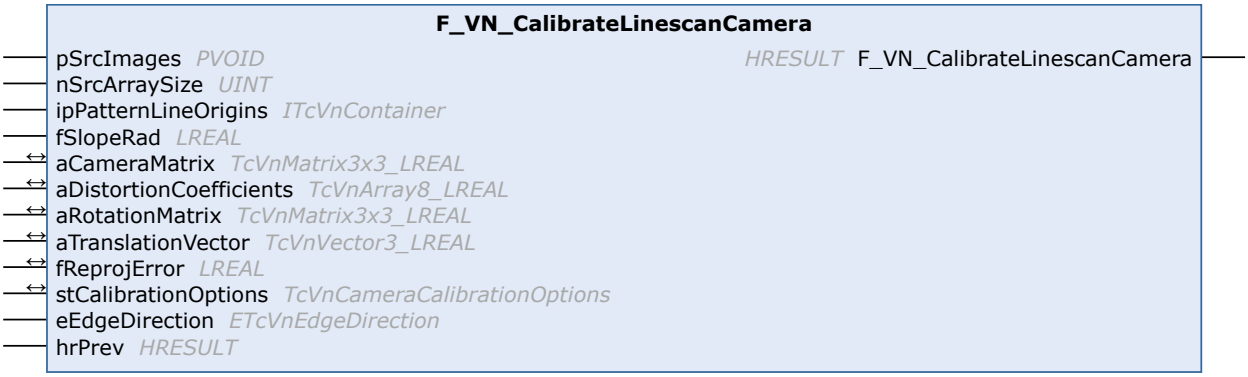
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.9 F_VN_CalibrateLinescanCamera



Compute the camera parameters (intrinsic + extrinsic) by evaluating images containing a line calibration pattern. The pattern must consist of alternating vertical and diagonal lines, with each diagonal having the same constant slope: |\|\|...\|. The results are only valid for the x-direction of a line scan image. To get accurate x-values for a transformation into world coordinates, the y-coordinate must be set to half the sensor width and so equal to the cy-value of the camera matrix. For other y-values, the results are approximated and can be inaccurate. The extrinsic parameters are computed for the first image in the array.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_CalibrateLinescanCamera : HRESULT
VAR_INPUT
    pSrcImages              : PVOID;
    nSrcArraySize           : UINT;
    ipPatternLineOrigins    : ITcVnContainer;
    fSlopeRad               : LREAL;
END_VAR
VAR_IN_OUT
    aCameraMatrix           : TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : TcVnArray8_LREAL;
    aRotationMatrix         : TcVnMatrix3x3_LREAL;
    aTranslationVector      : TcVnVector3_LREAL;
    fReprojError            : LREAL;
    stCalibrationOptions    : TcVnCameraCalibrationOptions;
END_VAR
VAR_INPUT
    eEdgeDirection          : ETcVnEdgeDirection;
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
pSrcImages	PVOID	Pointer to an array of 1 or more source images (1 channel). If the images have more than 1 row, each row must contain an image of the same pattern position.
nSrcArraySize	UINT	pSrcImages array size
ipPatternLineOrigins	ITcVnContainer [▶ 385]	X position of the pattern line origins (ContainerType_Vector_REAL, usually same origin for vertical and following diagonal line)
fSlopeRad	LREAL	Slope of the diagonal line in rad (> 0, < PI/2), relative to the vertical line (i.e., 0 would be a vertical line, PI/2 a horizontal line)
eEdgeDirection	ETcVnEdgeDirection [▶ 210]	Specification of the edge direction to search for
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 173]	Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 173]	Returns the translation vector
fReprojError	LREAL	Returns the reprojection error
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 240]	Calibration options (recommended to set bFixAspectRatio, bFixPrincipalPoint, bZeroTangentDist)

Return value

HRESULT [▶ 150]

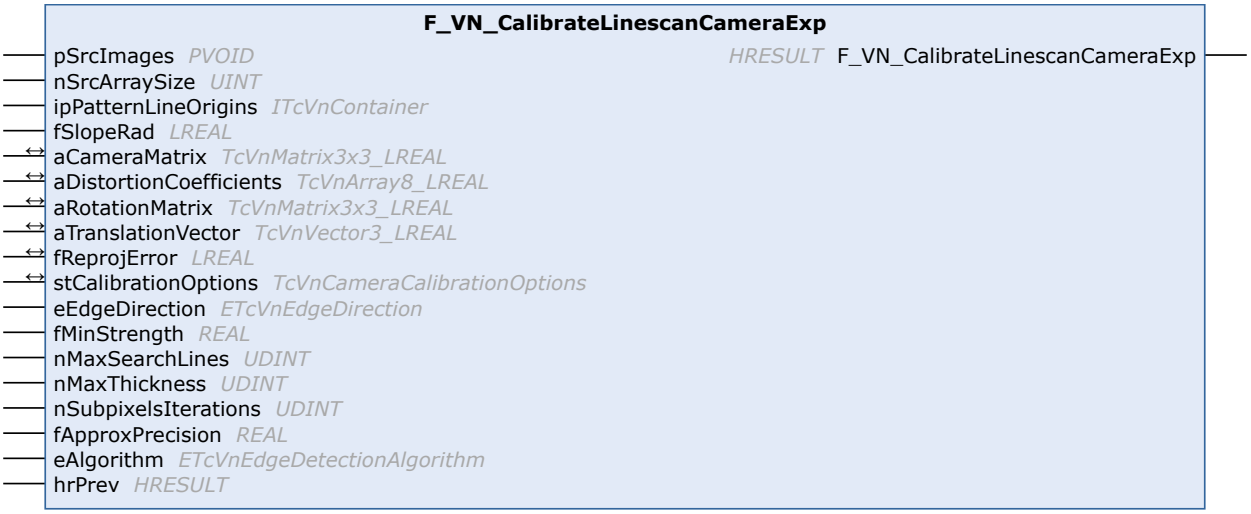
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.10 F_VN_CalibrateLinescanCameraExp



Compute the camera parameters (intrinsic + extrinsic) by evaluating images containing a line calibration pattern. The pattern must consist of alternating vertical and diagonal lines, with each diagonal having the same constant slope: |\|\|...\|. The results are only valid for the x-direction of a line scan image. To get accurate x-values for a transformation into world coordinates, the y-coordinate must be set to half the sensor width and so equal to the cy-value of the camera matrix. For other y-values, the results are approximated and can be inaccurate. The extrinsic parameters are computed for the first image in the array. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_CalibrateLinescanCameraExp : HRESULT
VAR_INPUT
    pSrcImages              : PVOID;
    nSrcArraySize           : UINT;
    ipPatternLineOrigins    : ITcVnContainer;
    fSlopeRad               : LREAL;
END_VAR
VAR_IN_OUT
    aCameraMatrix           : TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : TcVnArray8_LREAL;
    aRotationMatrix         : TcVnMatrix3x3_LREAL;
    aTranslationVector      : TcVnVector3_LREAL;
    fReprojError            : LREAL;
    stCalibrationOptions    : TcVnCameraCalibrationOptions;
END_VAR
VAR_INPUT
    eEdgeDirection          : ETcVnEdgeDirection;
    fMinStrength            : REAL;
    nMaxSearchLines         : UDINT;
    nMaxThickness           : UDINT;
    nSubpixelsIterations    : UDINT;
    fApproxPrecision        : REAL;
    eAlgorithm              : ETcVnEdgeDetectionAlgorithm;
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
pSrcImages	PVOID	Pointer to an array of 1 or more source images (1 channel). If the images have more than 1 row, each row must contain an image of the same pattern position.
nSrcArraySize	UINT	pSrcImages array size
ipPatternLineOrigins	ITcVnContainer [▶ 385]	X position of the pattern line origins (ContainerType_Vector_REAL, usually same origin for vertical and following diagonal line)
fSlopeRad	LREAL	Slope of the diagonal line in rad (> 0, < PI/2), relative to the vertical line (i.e., 0 would be a vertical line, PI/2 a horizontal line)
eEdgeDirection	ETcVnEdgeDirection [▶ 210]	Specification of the edge direction to search for
fMinStrength	REAL	Specification of the minimum strength (intensity difference) of the edge to search for
nMaxSearchLines	UDINT	Maximum number of search lines (equally distributed over the image height, at most 1 searchline per image row)
nMaxThickness	UDINT	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels
nSubpixelsIterations	UDINT	Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)
fApproxPrecision	REAL	Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)
eAlgorithm	ETcVnEdgeDetectionAlgorithm [▶ 209]	Selection of the edge detection algorithm
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 173]	Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 173]	Returns the translation vector
fReprojError	LREAL	Returns the reprojection error
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 240]	Calibration options (recommended to set bFixAspectRatio, bFixPrincipalPoint, bZeroTangentDist)

Return value

HRESULT [▶ 150]

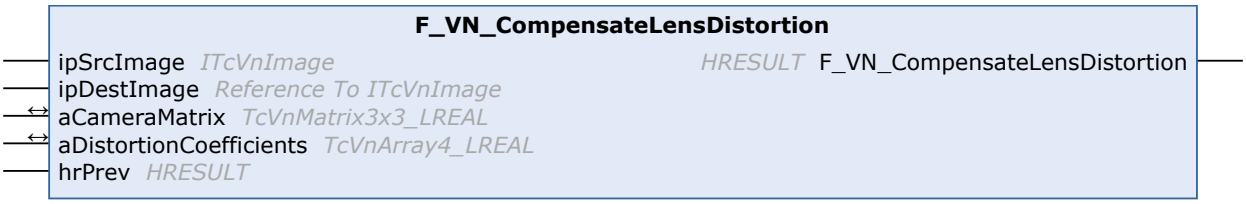
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.11 F_VN_CompensateLensDistortion



Transforms an image to compensate the lens distortion.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_CompensateLensDistortion : HRESULT
VAR_INPUT
    ipSrcImage              : ITcVnImage;
    ipDestImage             : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    aCameraMatrix           : TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : TcVnArray4_LREAL;
END_VAR
VAR_INPUT
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray4_LREAL [▶ 173]	Lens distortion coefficients [k1, k2, p1, p2]

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.12 F_VN_CompensateLensDistortionExp1

Transforms an image to compensate the lens distortion. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_CompensateLensDistortionExp1 : HRESULT
VAR_INPUT
    ipSrcImage              : ITcVnImage;
    ipDestImage             : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    aCameraMatrix           : TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : TcVnArray8_LREAL;
END_VAR
VAR_INPUT
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 173]	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.13 F_VN_CompensateLensDistortionExp2

Transforms an image to compensate the lens distortion. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_CompensateLensDistortionExp2 : HRESULT
VAR_INPUT
    ipSrcImage              : ITcVnImage;
    ipDestImage             : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    aCameraMatrix           : TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : TcVnArray8_LREAL;
    aNewCameraMatrix        : TcVnMatrix3x3_LREAL;
END_VAR
VAR_INPUT
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 173]	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aNewCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Allows additional scaling and shifting of the result image

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.14 F_VN_CompensateLensDistortionForPoints

Transforms point coordinates to compensate the lens distortion.

Syntax

Definition:

```
FUNCTION F_VN_CompensateLensDistortionForPoints : HRESULT
VAR_INPUT
    ipSrcPoints             : ITcVnContainer;
    ipDestPoints            : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    aCameraMatrix           : TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : TcVnArray4_LREAL;
END_VAR
VAR_INPUT
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcPoints	ITcVnContainer [▶ 385]	Container with source point coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_DINT)
ipDestPoints	Reference To ITcVnContainer [▶ 385]	Returns a container with the transformed point coordinates (ContainerType_Vector_TcVnPoint2_REAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray4_LREAL [▶ 173]	Lens distortion coefficients [k1, k2, p1, p2]

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.15 F_VN_CompensateLensDistortionForPointsExp1

Transforms point coordinates to compensate the lens distortion. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_CompensateLensDistortionForPointsExp1 : HRESULT
VAR_INPUT
    ipSrcPoints             : ITcVnContainer;
    ipDestPoints            : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    aCameraMatrix           : TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : TcVnArray8_LREAL;
END_VAR
VAR_INPUT
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcPoints	ITcVnContainer [▶ 385]	Container with source point coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_DINT)
ipDestPoints	Reference To ITcVnContainer [▶ 385]	Returns a container with the transformed point coordinates (ContainerType_Vector_TcVnPoint2_REAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 173]	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.16 F_VN_CompensateLensDistortionForPointsExp2

Transforms point coordinates to compensate the lens distortion. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_CompensateLensDistortionForPointsExp2 : HRESULT
VAR_INPUT
    ipSrcPoints             : ITcVnContainer;
    ipDestPoints            : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    aCameraMatrix           : TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : TcVnArray8_LREAL;
    aNewCameraMatrix        : TcVnMatrix3x3_LREAL;
END_VAR
VAR_INPUT
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcPoints	ITcVnContainer [▶ 385]	Container with source point coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_DINT)
ipDestPoints	Reference To ITcVnContainer [▶ 385]	Returns a container with the transformed point coordinates (ContainerType_Vector_TcVnPoint2_REAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 173]	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aNewCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Allows additional shifting of the result point coordinates

Return value

HRESULT [▶ 150]

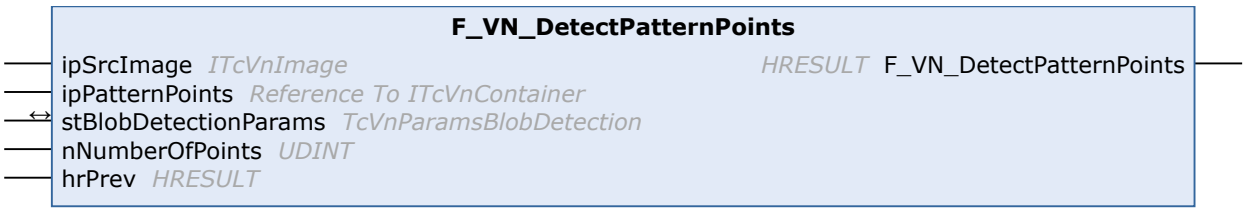
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.17 F_VN_DetectPatternPoints



Detects calibration pattern points (circles) within the provided image.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_DetectPatternPoints : HRESULT
VAR_INPUT
    ipSrcImage            : ITcVnImage;
    ipPatternPoints       : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    stBlobDetectionParams : TcVnParamsBlobDetection;
END_VAR
```

```
VAR_INPUT
    nNumberOfPoints       : UDINT;
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, 1 channel)
ipPatternPoints	Reference To ITcVnContainer [▶ 385]	Returns the pattern point positions (ContainerType_Vector_TcVnPoint2_REAL; Non-zero interface pointers are reused.)
nNumberOfPoints	UDINT	Expected number of pattern points
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stBlobDetectionParams	TcVnParamsBlobDetection [▶ 247]	Parameters to detect and filter contours.

Return value

HRESULT [▶ 150]

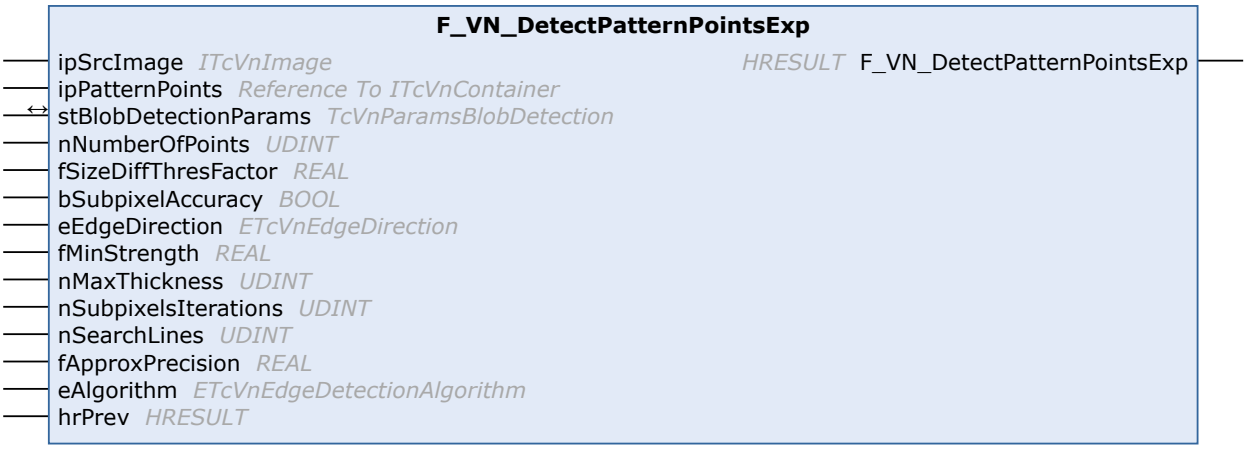
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.18 F_VN_DetectPatternPointsExp



Detects calibration pattern points (circles) within the provided image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_DetectPatternPointsExp : HRESULT
VAR_INPUT
    ipSrcImage            : ITcVnImage;
    ipPatternPoints       : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    stBlobDetectionParams : TcVnParamsBlobDetection;
END_VAR
VAR_INPUT
    nNumberOfPoints       : UDINT;
    fSizeDiffThresFactor  : REAL;
    bSubpixelAccuracy     : BOOL;
    eEdgeDirection        : ETcVnEdgeDirection;
    fMinStrength          : REAL;
    nMaxThickness         : UDINT;
    nSubpixelsIterations  : UDINT;
    nSearchLines          : UDINT;
    fApproxPrecision      : REAL;
    eAlgorithm            : ETcVnEdgeDetectionAlgorithm;
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, 1 channel)
ipPatternPoints	Reference To ITcVnContainer [▶ 385]	Returns the pattern point positions (ContainerType_Vector_TcVnPoint2_REAL; Non-zero interface pointers are reused.)
nNumberOfPoints	UDINT	Expected number of pattern points
fSizeDiffThresFactor	REAL	Threshold for filtering contours by average size
bSubpixelAccuracy	BOOL	If true, the pattern points are detected with subpixel accuracy
eEdgeDirection	ETcVnEdgeDirection [▶ 210]	Specification of the edge direction to search for (from center to outside circle)
fMinStrength	REAL	Specification of the minimum strength (intensity difference) of the edge to search for
nMaxThickness	UDINT	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels
nSubpixelsIterations	UDINT	Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)
nSearchLines	UDINT	Specifies the amount of search lines, which are equally distributed in all directions (must be >= 8 and a multiple of 4)
fApproxPrecision	REAL	Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)
eAlgorithm	ETcVnEdgeDetectionAlgorithm [▶ 209]	Selection of the edge detection algorithm
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stBlobDetectionParams	TcVnParamsBlobDetection [▶ 247]	Parameters to detect and filter contours.

Return value

HRESULT [▶ 150]

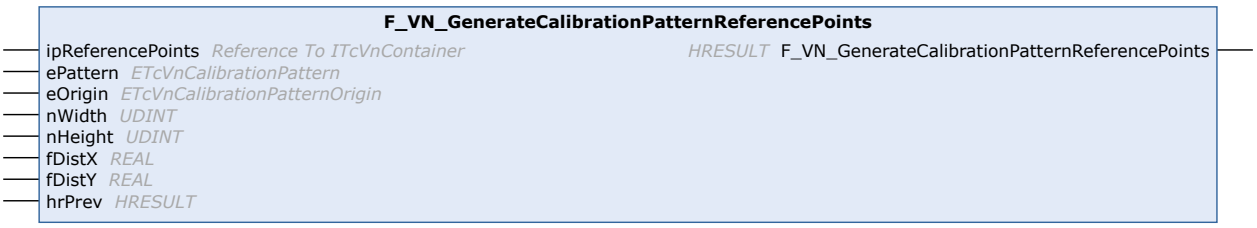
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.19 F_VN_GenerateCalibrationPatternReferencePoints



Generate reference points for common calibration patterns.

Syntax

Definition:

```
FUNCTION F_VN_GenerateCalibrationPatternReferencePoints : HRESULT
VAR_INPUT
    ipReferencePoints : Reference To ITcVnContainer;
    ePattern          : ETcVnCalibrationPattern;
    eOrigin           : ETcVnCalibrationPatternOrigin;
    nWidth            : UDINT;
    nHeight           : UDINT;
    fDistX            : REAL;
    fDistY            : REAL;
    hrPrev            : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipReferencePoints	Reference To ITcVnContainer [▶ 385]	Returns the generated reference points (ContainerType_Vector_TcVnPoint3_REAL)
ePattern	ETcVnCalibrationPattern [▶ 179]	Calibration pattern type
eOrigin	ETcVnCalibrationPatternOrigin [▶ 179]	Calibration pattern origin
nWidth	UDINT	Pattern width, i.e. number of pattern points in x direction
nHeight	UDINT	Pattern height, i.e. number of pattern points in y direction
fDistX	REAL	Distance between pattern points in x direction
fDistY	REAL	Distance between pattern points in y direction
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

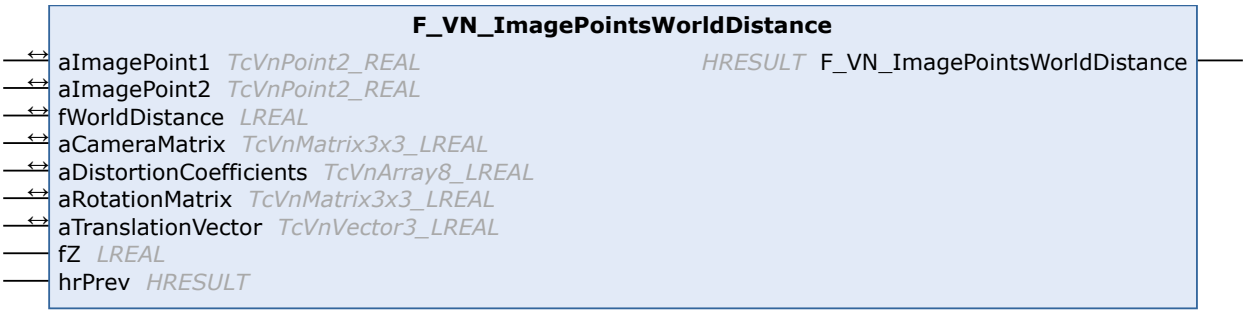
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.20 F_VN_ImagePointsWorldDistance



Computes the distance in the world coordinate system between two points (in the same world coordinate z layer) given in image coordinates.

Syntax

Definition:

```
FUNCTION F_VN_ImagePointsWorldDistance : HRESULT
VAR_IN_OUT
    aImagePoint1            : TcVnPoint2_REAL;
    aImagePoint2            : TcVnPoint2_REAL;
    fWorldDistance          : LREAL;
    aCameraMatrix           : TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : TcVnArray8_LREAL;
    aRotationMatrix         : TcVnMatrix3x3_LREAL;
    aTranslationVector      : TcVnVector3_LREAL;
END_VAR
```

```
VAR_INPUT
    fZ                      : LREAL;
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fZ	LREAL	z coordinate (world coordinate system) of the given points (0 would be at (toplevel) calibration pattern)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aImagePoint1	TcVnPoint2_REAL [▶ 171]	First point in the image coordinate system
aImagePoint2	TcVnPoint2_REAL [▶ 171]	Second point in the image coordinate system
fWorldDistance	LREAL	Returns the distance between aImagePoint1 and aImagePoint2 in the world coordinate system
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	3x3 camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 173]	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 173]	Translation vector

Return value

HRESULT [▶ 150]

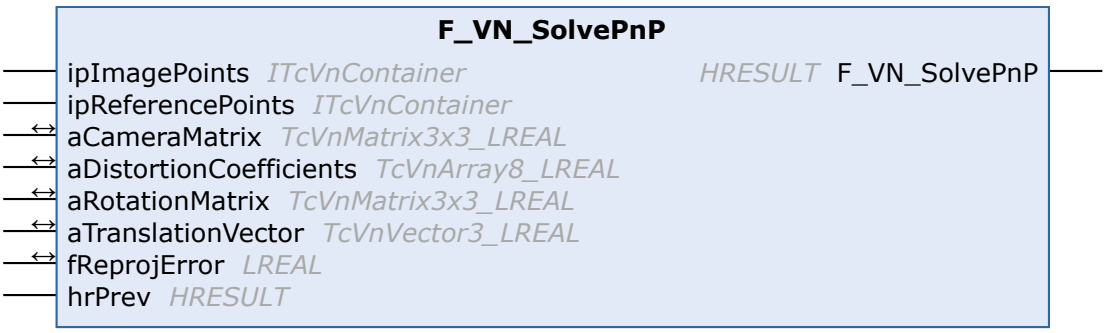
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.21 F_VN_SolvePnP



Computes the rotation matrix and translation vector from 3D-2D point correspondences.

Syntax

Definition:

```
FUNCTION F_VN_SolvePnP : HRESULT
VAR_INPUT
    ipImagePoints           : ITcVnContainer;
    ipReferencePoints       : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aCameraMatrix           : TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : TcVnArray8_LREAL;
    aRotationMatrix         : TcVnMatrix3x3_LREAL;
    aTranslationVector      : TcVnVector3_LREAL;
    fReprojError            : LREAL;
END_VAR
VAR_INPUT
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImagePoints	ITcVnContainer [▶ 385]	Image points (ContainerType_Vector_TcVnPoint2_REAL, requires at least 4 points in the same order as in ipReferencePoints)
ipReferencePoints	ITcVnContainer [▶ 385]	Reference world points (ContainerType_Vector_TcVnPoint3_REAL, same number of points as in ipImagePoints)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Camera matrix (e.g. computed with one of the CalibrateCamera functions)
aDistortionCoefficients	TcVnArray8_LREAL [▶ 173]	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6] (e.g. computed with one of the CalibrateCamera functions)
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 173]	Returns the translation vector
fReprojError	LREAL	Returns the reprojection error

Return value

HRESULT [▶ 150]

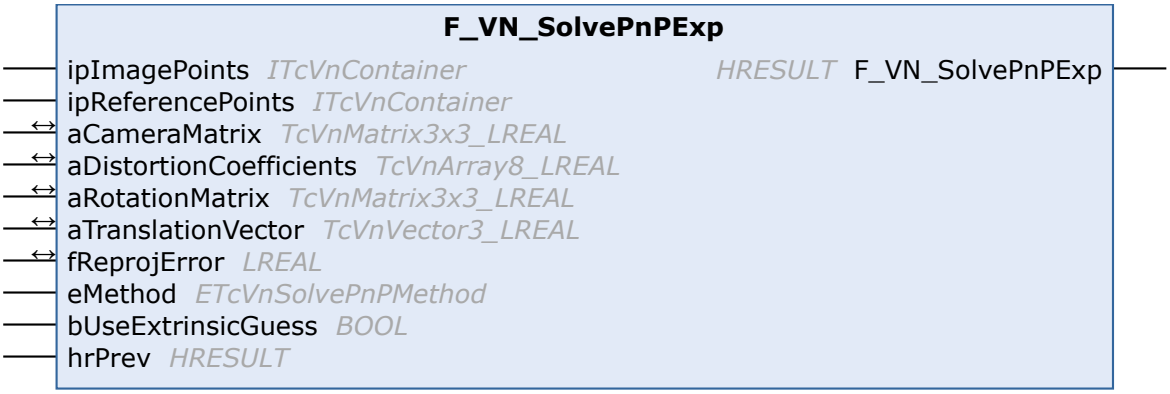
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.22 F_VN_SolvePnPExp



Computes the rotation matrix and translation vector from 3D-2D point correspondences. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_SolvePnPExp : HRESULT
VAR_INPUT
    ipImagePoints           : ITcVnContainer;
    ipReferencePoints       : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aCameraMatrix           : TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : TcVnArray8_LREAL;
    aRotationMatrix         : TcVnMatrix3x3_LREAL;
    aTranslationVector      : TcVnVector3_LREAL;
    fReprojError            : LREAL;
END_VAR
VAR_INPUT
    eMethod                 : ETcVnSolvePnPMethod;
    bUseExtrinsicGuess      : BOOL;
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImagePoints	ITcVnContainer [▶ 385]	Image points (ContainerType_Vector_TcVnPoint2_REAL, requires at least 4 points in the same order as in ipReferencePoints)
ipReferencePoints	ITcVnContainer [▶ 385]	Reference world points (ContainerType_Vector_TcVnPoint3_REAL, same number of points as in ipImagePoints)
eMethod	ETcVnSolvePnPMethod [▶ 228]	Solve PnP method
bUseExtrinsicGuess	BOOL	If true, the provided rotation matrix and translation vector are used as an initial guess for further optimization (only supported for the SPM_ITERATIVE method).
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Camera matrix (e.g. computed with one of the CalibrateCamera functions)
aDistortionCoefficients	TcVnArray8_LREAL [▶ 173]	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6] (e.g. computed with one of the CalibrateCamera functions)
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 173]	Returns the translation vector
fReprojError	LREAL	Returns the reprojection error

Return value

HRESULT [▶ 150]

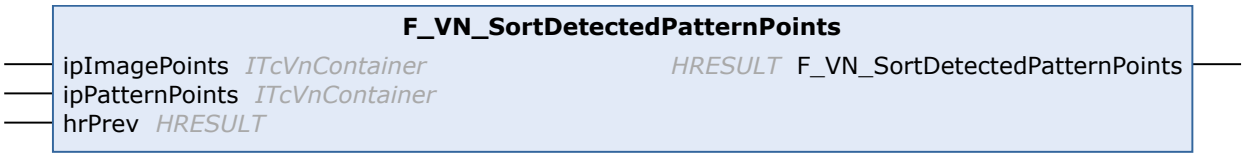
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.23 F_VN_SortDetectedPatternPoints



Sort the detected pattern points according to the relative positions of the calibration pattern points.

Syntax

Definition:

```
FUNCTION F_VN_SortDetectedPatternPoints : HRESULT
VAR_INPUT
    ipImagePoints   : ITcVnContainer;
    ipPatternPoints : ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImagePoints	ITcVnContainer [▶ 385]	Detected pattern point positions in the image coordinate system (will be sorted by this function, ContainerType_Vector_TcVnPoint2_REAL)
ipPatternPoints	ITcVnContainer [▶ 385]	The pattern points in the calibration pattern coordinate system, used as a reference to sort ipImagePoints (ContainerType_Vector_TcVnPoint3_REAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

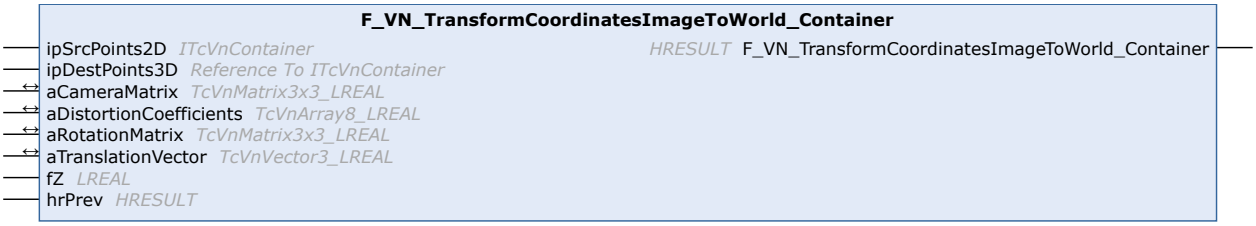
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.24 F_VN_TransformCoordinatesImageToWorld_Container



Transform 2D image point coordinates to 3D world coordinates (using intrinsic + extrinsic calibration results).

Syntax

Definition:

```
FUNCTION F_VN_TransformCoordinatesImageToWorld_Container : HRESULT
VAR_INPUT
    ipSrcPoints2D           : ITcVnContainer;
    ipDestPoints3D          : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    aCameraMatrix           : TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : TcVnArray8_LREAL;
    aRotationMatrix         : TcVnMatrix3x3_LREAL;
    aTranslationVector      : TcVnVector3_LREAL;
END_VAR
VAR_INPUT
    fZ                      : LREAL;
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcPoints2D	ITcVnContainer [▶ 385]	Container with 2D source points (TcVnPoint2_DINT or TcVnPoint2_REAL or TcVnPoint2_LREAL)
ipDestPoints3D	Reference To ITcVnContainer [▶ 385]	Returns the transformed 3D points (TcVnPoint3_LREAL)
fZ	LREAL	z coordinate (world coordinate system) of the transformed points (0 would be at (toplevel) calibration pattern)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	3x3 camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 173]	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 173]	Translation vector

Return value

HRESULT [▶ 150]

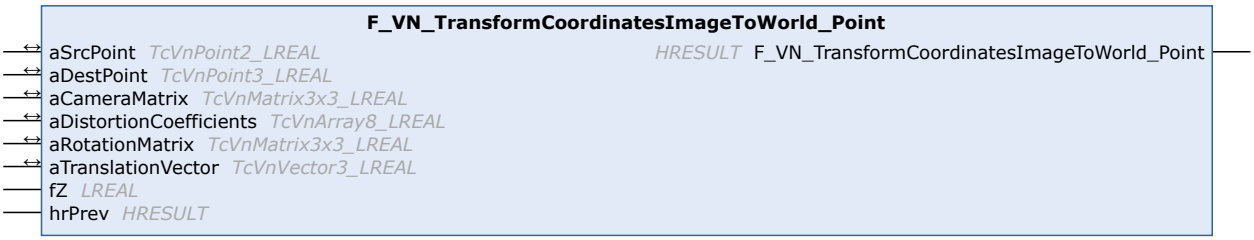
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.25 F_VN_TransformCoordinatesImageToWorld_Point



Transform 2D image point coordinate to 3D world coordinate (using intrinsic + extrinsic calibration results).

Syntax

Definition:

```
FUNCTION F_VN_TransformCoordinatesImageToWorld_Point : HRESULT
VAR_IN_OUT
    aSrcPoint               : TcVnPoint2_LREAL;
    aDestPoint              : TcVnPoint3_LREAL;
    aCameraMatrix           : TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : TcVnArray8_LREAL;
    aRotationMatrix         : TcVnMatrix3x3_LREAL;
    aTranslationVector      : TcVnVector3_LREAL;
END_VAR
VAR_INPUT
    fZ                      : LREAL;
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fZ	LREAL	z coordinate (world coordinate system) of the transformed points (0 would be at (toplevel) calibration pattern)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aSrcPoint	TcVnPoint2_LREAL [▶ 171]	Point in the image coordinate system
aDestPoint	TcVnPoint3_LREAL [▶ 171]	Returns the point in the world coordinate system
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	3x3 camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 173]	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 173]	Translation vector

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.26 F_VN_TransformCoordinatesPlanar_Container

Compute real-world coordinates for given image points or vice versa.

Syntax

Definition:

```
FUNCTION F_VN_TransformCoordinatesPlanar_Container : HRESULT
VAR_INPUT
    ipSrcPoints           : ITcVnContainer;
    ipDestPoints          : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    aTransformationMatrix : TcVnMatrix3x3_LREAL;
END_VAR
```

```
VAR_INPUT
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcPoints	ITcVnContainer [▶ 385]	Container with 2D source points (TcVnPoint2_DINT or TcVnPoint2_REAL or TcVnPoint2_LREAL)
ipDestPoints	Reference To ITcVnContainer [▶ 385]	Returns the transformed 2D points (TcVnPoint2_LREAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Transformation matrix obtained with VnCalibrateCameraPlanar() or its inverse matrix obtained with VnInvertMatrix3x3()

Return value

HRESULT [▶ 150]

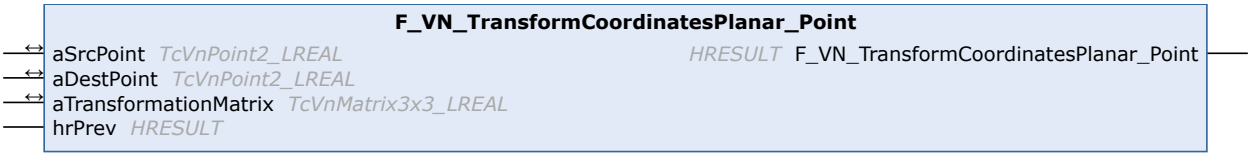
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.27 F_VN_TransformCoordinatesPlanar_Point



Compute real-world coordinates for a given image point or vice versa.

Syntax

Definition:

```
FUNCTION F_VN_TransformCoordinatesPlanar_Point : HRESULT
VAR_IN_OUT
    aSrcPoint             : TcVnPoint2_LREAL;
    aDestPoint            : TcVnPoint2_LREAL;
    aTransformationMatrix : TcVnMatrix3x3_LREAL;
END_VAR
VAR_INPUT
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aSrcPoint	TcVnPoint2_LREAL [▶ 171]	Point in the source coordinate system
aDestPoint	TcVnPoint2_LREAL [▶ 171]	Returns point in the destination coordinate system
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Transformation matrix obtained with VnCalibrateCameraPlanar() or its inverse matrix obtained with VnInvertMatrix3x3()

Return value

HRESULT [▶ 150]

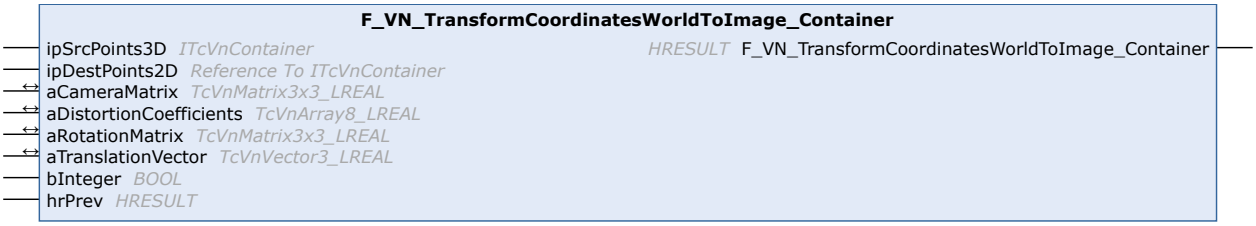
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.28 F_VN_TransformCoordinatesWorldToImage_Container



Transform 3D world coordinates to 2D image coordinates (using intrinsic + extrinsic calibration results).

Syntax

Definition:

```
FUNCTION F_VN_TransformCoordinatesWorldToImage_Container : HRESULT
VAR_INPUT
    ipSrcPoints3D           : ITcVnContainer;
    ipDestPoints2D          : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    aCameraMatrix           : TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : TcVnArray8_LREAL;
    aRotationMatrix         : TcVnMatrix3x3_LREAL;
    aTranslationVector      : TcVnVector3_LREAL;
END_VAR
VAR_INPUT
    bInteger                : BOOL;
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcPoints3D	ITcVnContainer [▶ 385]	Container with 3D source points (TcVnPoint3_LREAL)
ipDestPoints2D	Reference To ITcVnContainer [▶ 385]	Returns the transformed 2D points (TcVnPoint2_REAL or TcVnPoint2_DINT, depending on bInteger)
bInteger	BOOL	Selects whether the function returns a container of TcVnPoint2_DINT or TcVnPoint2_REAL
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	3x3 camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 173]	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 173]	Translation vector

Return value

HRESULT [▶ 150]

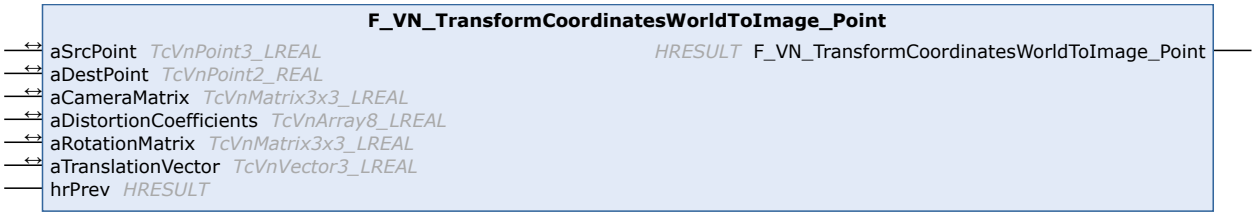
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.1.29 F_VN_TransformCoordinatesWorldToImage_Point



Transform 3D world coordinates to 2D image coordinates (using intrinsic + extrinsic calibration results).

Syntax

Definition:

```
FUNCTION F_VN_TransformCoordinatesWorldToImage_Point : HRESULT
VAR_IN_OUT
    aSrcPoint               : TcVnPoint3_LREAL;
    aDestPoint              : TcVnPoint2_REAL;
    aCameraMatrix           : TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : TcVnArray8_LREAL;
```

```
    aRotationMatrix         : TcVnMatrix3x3_LREAL;
    aTranslationVector      : TcVnVector3_LREAL;
END_VAR
VAR_INPUT
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aSrcPoint	TcVnPoint3_LREAL [▶ 171]	3D source point (world coordinate system)
aDestPoint	TcVnPoint2_REAL [▶ 171]	Returns the transformed 2D point (image coordinate system)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	3x3 camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 173]	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 173]	Translation vector

Return value

HRESULT [▶ 150]

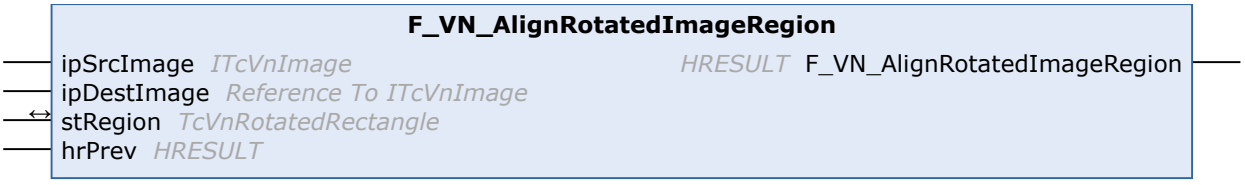
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.2 F_VN_AlignRotatedImageRegion



Copies the provided image region and aligns it to the image axes.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_AlignRotatedImageRegion : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    stRegion    : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRegion	TcVnRotatedRectangle [▶ 258]	Image region to be aligned

Return value

HRESULT [▶ 150]

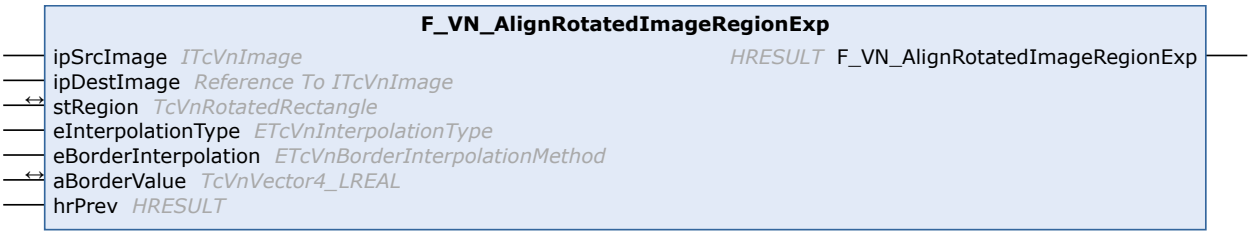
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.3 F_VN_AlignRotatedImageRegionExp



Copies the provided image region and aligns it to the image axes. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_AlignRotatedImageRegionExp : HRESULT
VAR_INPUT
    ipSrcImage           : ITcVnImage;
```

```
    ipDestImage          : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    stRegion             : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    eInterpolationType   : ETcVnInterpolationType;
    eBorderInterpolation : ETcVnBorderInterpolationMethod;
END_VAR
VAR_IN_OUT
    aBorderValue         : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev               : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate image will be created if required.)
eInterpolationType	ETcVnInterpolationType [▶ 217]	Interpolation method
eBorderInterpolation	ETcVnBorderInterpolationMethod [▶ 177]	Border interpolation method (ISOLATED not supported)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRegion	TcVnRotatedRectangle [▶ 258]	Image region to be aligned
aBorderValue	TcVnVector4_LREAL [▶ 173]	Border value, if CONSTANT is used

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.4 F_VN_ApplyRotationToAffineTransformation

Apply a rotation to an existing 2D affine transformation matrix.

Syntax

Definition:

```
FUNCTION F_VN_ApplyRotationToAffineTransformation : HRESULT
VAR_IN_OUT
    aTransformationMatrix : TcVnMatrix2x3_LREAL;
END_VAR
VAR_INPUT
    fAngle                : LREAL;
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fAngle	LREAL	Angle in radians (positive means counter-clockwise)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 173]	Affine transformation matrix to which the rotation is applied

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.5 F_VN_ApplyRotationToAffineTransformationExp

Apply a rotation to an existing 2D affine transformation matrix. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_ApplyRotationToAffineTransformationExp : HRESULT
VAR_IN_OUT
    aTransformationMatrix : TcVnMatrix2x3_LREAL;
END_VAR
VAR_INPUT
    fAngle                : LREAL;
```

```
    bUsePreMultiplication : BOOL;
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fAngle	LREAL	Angle in radians (positive means counter-clockwise)
bUsePreMultiplication	BOOL	Select if pre- or post-multiplication should be used
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 173]	Affine transformation matrix to which the rotation is applied

Return value

HRESULT [▶ 150]

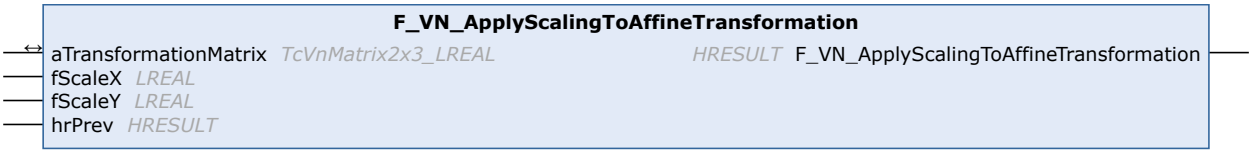
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.6 F_VN_ApplyScalingToAffineTransformation



Apply a scaling to an existing 2D affine transformation matrix.

Syntax

Definition:

```
FUNCTION F_VN_ApplyScalingToAffineTransformation : HRESULT
VAR_IN_OUT
    aTransformationMatrix : TcVnMatrix2x3_LREAL;
END_VAR
VAR_INPUT
    fScaleX               : LREAL;
    fScaleY               : LREAL;
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fScaleX	LREAL	Scaling factor in x direction
fScaleY	LREAL	Scaling factor in y direction
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 173]	Affine transformation matrix to which the scaling is applied

Return value

HRESULT [▶ 150]

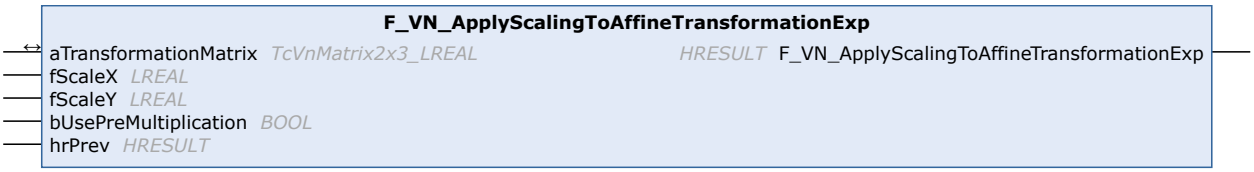
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.7 F_VN_ApplyScalingToAffineTransformationExp



Apply a scaling to an existing 2D affine transformation matrix. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_ApplyScalingToAffineTransformationExp : HRESULT
VAR_IN_OUT
    aTransformationMatrix : TcVnMatrix2x3_LREAL;
END_VAR
VAR_INPUT
    fScaleX               : LREAL;
    fScaleY               : LREAL;
    bUsePreMultiplication : BOOL;
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fScaleX	LREAL	Scaling factor in x direction
fScaleY	LREAL	Scaling factor in y direction
bUsePreMultiplication	BOOL	Select if pre- or post-multiplication should be used
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 173]	Affine transformation matrix to which the scaling is applied

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.8 F_VN_ApplyTranslationToAffineTransformation

Apply a translation to an existing 2D affine transformation matrix.

Syntax

Definition:

```
FUNCTION F_VN_ApplyTranslationToAffineTransformation : HRESULT
VAR_IN_OUT
    aTransformationMatrix : TcVnMatrix2x3_LREAL;
END_VAR
VAR_INPUT
    fDeltaX               : LREAL;
    fDeltaY               : LREAL;
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fDeltaX	LREAL	Translation in x direction
fDeltaY	LREAL	Translation in y direction
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 173]	Affine transformation matrix to which the translation is applied

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.9 F_VN_ApplyTranslationToAffineTransformationExp

Apply a translation to an existing 2D affine transformation matrix. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_ApplyTranslationToAffineTransformationExp : HRESULT
VAR_IN_OUT
    aTransformationMatrix : TcVnMatrix2x3_LREAL;
END_VAR
VAR_INPUT
    fDeltaX               : LREAL;
    fDeltaY               : LREAL;
    bUsePreMultiplication : BOOL;
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fDeltaX	LREAL	Translation in x direction
fDeltaY	LREAL	Translation in y direction
bUsePreMultiplication	BOOL	Select if pre- or post-multiplication should be used
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 173]	Affine transformation matrix to which the translation is applied

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.10 F_VN_ApplyYAxisInversionToAffineTransformation

Apply an inversion of the y-axis direction to an existing 2D affine transformation matrix.

Syntax

Definition:

```
FUNCTION F_VN_ApplyYAxisInversionToAffineTransformation : HRESULT
VAR_IN_OUT
    aTransformationMatrix : TcVnMatrix2x3_LREAL;
END_VAR
VAR_INPUT
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 173]	Affine transformation matrix to which the inversion of the y-axis direction is applied

Return value

HRESULT [▶ 150]

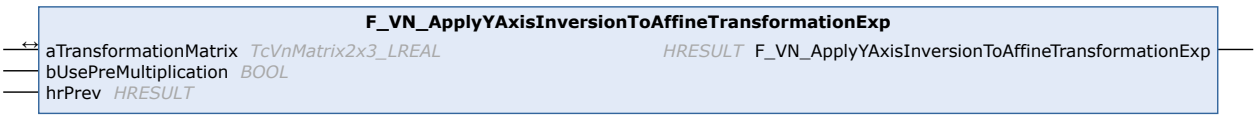
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.11 F_VN_ApplyYAxisInversionToAffineTransformationExp



Apply an inversion of the y-axis direction to an existing 2D affine transformation matrix. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_ApplyYAxisInversionToAffineTransformationExp : HRESULT
VAR_IN_OUT
    aTransformationMatrix : TcVnMatrix2x3_LREAL;
END_VAR
VAR_INPUT
    bUsePreMultiplication : BOOL;
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
bUsePreMultiplication	BOOL	Select if pre- or post-multiplication should be used
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 173]	Affine transformation matrix to which the inversion of the y-axis direction is applied

Return value

HRESULT [▶ 150]

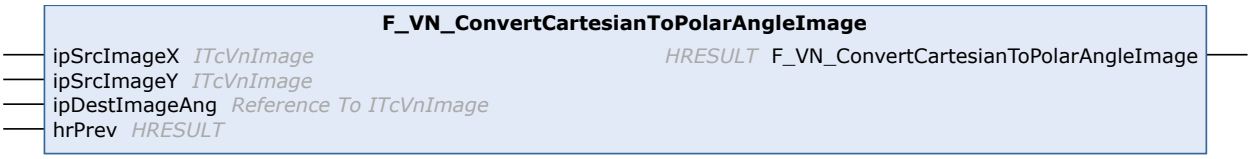
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.12 F_VN_ConvertCartesianToPolarAngleImage



Converts cartesian coordinates (x, y) to polar angle.

Syntax

Definition:

```
FUNCTION F_VN_ConvertCartesianToPolarAngleImage : HRESULT
VAR_INPUT
    ipSrcImageX    : ITcVnImage;
    ipSrcImageY    : ITcVnImage;
    ipDestImageAng : Reference To ITcVnImage;
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImageX	ITcVnImage [▶ 426]	Source image containing the x values (elements of type REAL or LREAL)
ipSrcImageY	ITcVnImage [▶ 426]	Source image containing the y values (Same element type as ipSrcImageX)
ipDestImageAng	Reference To ITcVnImage [▶ 426]	Destination image containing the angles (Same element type as source images. An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.13 F_VN_ConvertCartesianToPolarAngleImageExp

Converts cartesian coordinates (x, y) to polar angle. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_ConvertCartesianToPolarAngleImageExp : HRESULT
VAR_INPUT
    ipSrcImageX     : ITcVnImage;
    ipSrcImageY     : ITcVnImage;
    ipDestImageAng  : Reference To ITcVnImage;
    bAngleInDegrees : BOOL;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImageX	ITcVnImage [▶ 426]	Source image containing the x values (elements of type REAL or LREAL)
ipSrcImageY	ITcVnImage [▶ 426]	Source image containing the y values (Same element type as ipSrcImageX)
ipDestImageAng	Reference To ITcVnImage [▶ 426]	Destination image containing the angles (Same element type as source images. An appropriate destination image will be created if required.)
bAngleInDegrees	BOOL	Specifys, if the angles should be in degrees or radians
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

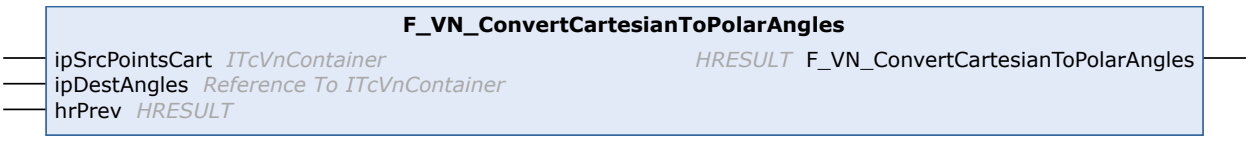
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.14 F_VN_ConvertCartesianToPolarAngles



Converts cartesian coordinates (x, y) to polar angle.

Syntax

Definition:

```
FUNCTION F_VN_ConvertCartesianToPolarAngles : HRESULT
VAR_INPUT
    ipSrcPointsCart : ITcVnContainer;
    ipDestAngles    : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcPointsCart	ITcVnContainer [▶ 385]	Container with cartesian points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestAngles	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the angles (ContainerType_Vector_REAL, if source points are of type TcVnPoint2_DINT or TcVnPoint2_REAL; else ContainerType_Vector_LREAL. Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

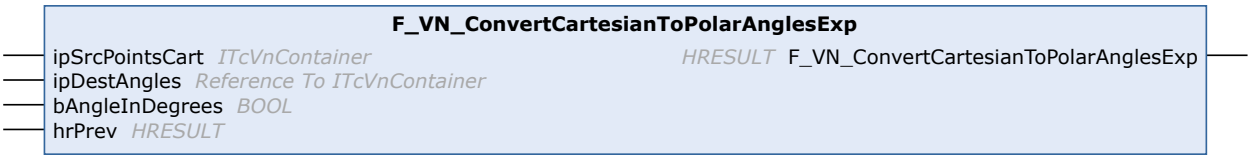
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.15 F_VN_ConvertCartesianToPolarAnglesExp



Converts cartesian coordinates (x, y) to polar angle. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_ConvertCartesianToPolarAnglesExp : HRESULT
VAR_INPUT
    ipSrcPointsCart : ITcVnContainer;
    ipDestAngles    : Reference To ITcVnContainer;
    bAngleInDegrees : BOOL;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcPointsCart	ITcVnContainer [▶ 385]	Container with cartesian points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestAngles	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the angles (ContainerType_Vector_REAL, if source points are of type TcVnPoint2_DINT or TcVnPoint2_REAL; else ContainerType_Vector_LREAL. Non-zero interface pointers are reused.)
bAngleInDegrees	BOOL	Specify, if the angles should be in degrees or radians
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

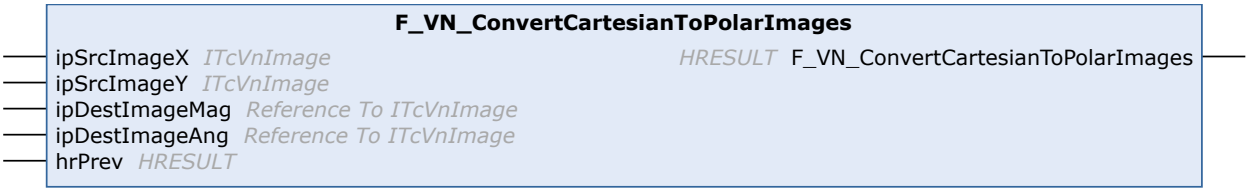
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.16 F_VN_ConvertCartesianToPolarImages



Converts cartesian coordinates (x, y) to polar coordinates (magnitude, angle).

Syntax

Definition:

```
FUNCTION F_VN_ConvertCartesianToPolarImages : HRESULT
VAR_INPUT
    ipSrcImageX    : ITcVnImage;
    ipSrcImageY    : ITcVnImage;
    ipDestImageMag : Reference To ITcVnImage;
    ipDestImageAng : Reference To ITcVnImage;
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImageX	ITcVnImage [▶ 426]	Source image containing the x values (elements of type REAL or LREAL)
ipSrcImageY	ITcVnImage [▶ 426]	Source image containing the y values (Same element type as ipSrcImageX)
ipDestImageMag	Reference To ITcVnImage [▶ 426]	Destination image containing the magnitudes (Same element type as source images. An appropriate destination image will be created if required.)
ipDestImageAng	Reference To ITcVnImage [▶ 426]	Destination image containing the angles (Same element type as source images. An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.17 F_VN_ConvertCartesianToPolarImagesExp

Converts cartesian coordinates (x, y) to polar coordinates (magnitude, angle). (expert function)

Syntax

Definition:

```
FUNCTION F_VN_ConvertCartesianToPolarImagesExp : HRESULT
VAR_INPUT
    ipSrcImageX     : ITcVnImage;
    ipSrcImageY     : ITcVnImage;
    ipDestImageMag  : Reference To ITcVnImage;
    ipDestImageAng  : Reference To ITcVnImage;
    bAngleInDegrees : BOOL;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImageX	ITcVnImage [▶ 426]	Source image containing the x values (elements of type REAL or LREAL)
ipSrcImageY	ITcVnImage [▶ 426]	Source image containing the y values (Same element type as ipSrcImageX)
ipDestImageMag	Reference To ITcVnImage [▶ 426]	Destination image containing the magnitudes (Same element type as source images. An appropriate destination image will be created if required.)
ipDestImageAng	Reference To ITcVnImage [▶ 426]	Destination image containing the angles (Same element type as source images. An appropriate destination image will be created if required.)
bAngleInDegrees	BOOL	Specifys, if the angles should be in degrees or radians
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

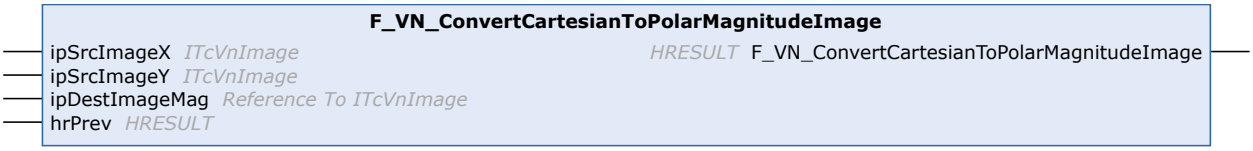
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.18 F_VN_ConvertCartesianToPolarMagnitudeImage



Converts cartesian coordinates (x, y) to polar magnitude.

Syntax

Definition:

```
FUNCTION F_VN_ConvertCartesianToPolarMagnitudeImage : HRESULT
VAR_INPUT
    ipSrcImageX    : ITcVnImage;
    ipSrcImageY    : ITcVnImage;
    ipDestImageMag : Reference To ITcVnImage;
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImageX	ITcVnImage [▶ 426]	Source image containing the x values (elements of type REAL or LREAL)
ipSrcImageY	ITcVnImage [▶ 426]	Source image containing the y values (Same element type as ipSrcImageX)
ipDestImageMag	Reference To ITcVnImage [▶ 426]	Destination image containing the magnitudes (Same element type as source images. An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

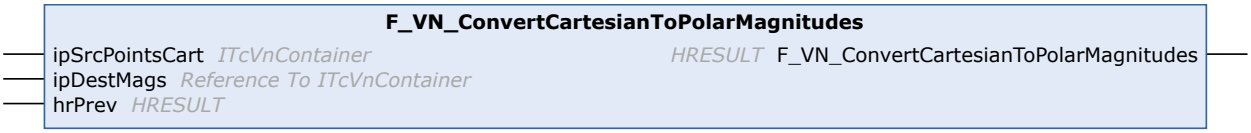
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.19 F_VN_ConvertCartesianToPolarMagnitudes



Converts cartesian coordinates (x, y) to polar magnitude.

Syntax

Definition:

```
FUNCTION F_VN_ConvertCartesianToPolarMagnitudes : HRESULT
VAR_INPUT
    ipSrcPointsCart : ITcVnContainer;
    ipDestMags      : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcPointsCart	ITcVnContainer [▶ 385]	Container with cartesian points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestMags	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the magnitudes (ContainerType_Vector_REAL, if source points are of type TcVnPoint2_DINT or TcVnPoint2_REAL; else ContainerType_Vector_LREAL. Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

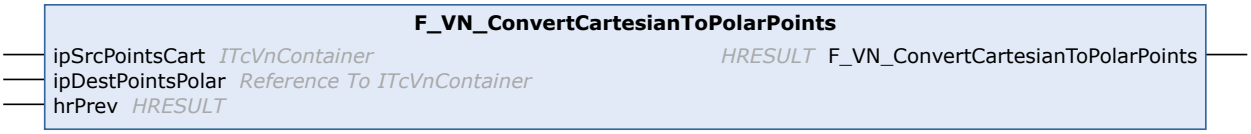
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.20 F_VN_ConvertCartesianToPolarPoints



Converts cartesian coordinates (x, y) to polar coordinates (magnitude, angle).

Syntax

Definition:

```
FUNCTION F_VN_ConvertCartesianToPolarPoints : HRESULT
VAR_INPUT
    ipSrcPointsCart   : ITcVnContainer;
    ipDestPointsPolar : Reference To ITcVnContainer;
    hrPrev            : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcPointsCart	ITcVnContainer [▶ 385]	Container with cartesian points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestPointsPolar	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the converted points in polar coordinates (ContainerType_Vector_TcVnPoint2_REAL, if source points are of type TcVnPoint2_DINT or TcVnPoint2_REAL; else ContainerType_Vector_TcVnPoint2_LREAL. Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.21 F_VN_ConvertCartesianToPolarPointsExp

Converts cartesian coordinates (x, y) to polar coordinates (magnitude, angle). (expert function)

Syntax

Definition:

```
FUNCTION F_VN_ConvertCartesianToPolarPointsExp : HRESULT
VAR_INPUT
    ipSrcPointsCart   : ITcVnContainer;
    ipDestPointsPolar : Reference To ITcVnContainer;
    bAngleInDegrees   : BOOL;
    hrPrev            : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcPointsCart	ITcVnContainer [▶ 385]	Container with cartesian points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestPointsPolar	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the converted points in polar coordinates (ContainerType_Vector_TcVnPoint2_REAL, if source points are of type TcVnPoint2_DINT or TcVnPoint2_REAL; else ContainerType_Vector_TcVnPoint2_LREAL. Non-zero interface pointers are reused.)
bAngleInDegrees	BOOL	Specifys, if the angles should be in degrees or radians
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

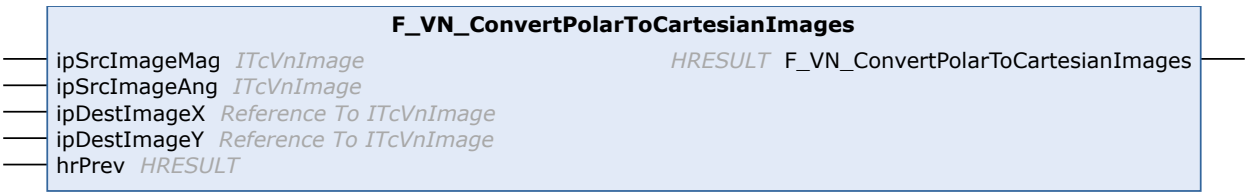
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.22 F_VN_ConvertPolarToCartesianImages



Converts polar coordinates (magnitude, angle) to cartesian coordinates (x, y).

Syntax

Definition:

```
FUNCTION F_VN_ConvertPolarToCartesianImages : HRESULT
VAR_INPUT
    ipSrcImageMag : ITcVnImage;
    ipSrcImageAng : ITcVnImage;
    ipDestImageX  : Reference To ITcVnImage;
    ipDestImageY  : Reference To ITcVnImage;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImageMag	ITcVnImage [▶ 426]	Source image containing the magnitudes (elements of type REAL or LREAL)
ipSrcImageAng	ITcVnImage [▶ 426]	Source image containing the angles (Same element type as ipSrcImageMag)
ipDestImageX	Reference To ITcVnImage [▶ 426]	Destination image containing the x values (Same element type as source images. An appropriate destination image will be created if required.)
ipDestImageY	Reference To ITcVnImage [▶ 426]	Destination image containing the y values (Same element type as source images. An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.23 F_VN_ConvertPolarToCartesianImagesExp

Converts polar coordinates (magnitude, angle) to cartesian coordinates (x, y). (expert function)

Syntax

Definition:

```
FUNCTION F_VN_ConvertPolarToCartesianImagesExp : HRESULT
VAR_INPUT
    ipSrcImageMag   : ITcVnImage;
    ipSrcImageAng   : ITcVnImage;
    ipDestImageX    : Reference To ITcVnImage;
    ipDestImageY    : Reference To ITcVnImage;
    bAngleInDegrees : BOOL;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImageMag	ITcVnImage [▶ 426]	Source image containing the magnitudes (elements of type REAL or LREAL)
ipSrcImageAng	ITcVnImage [▶ 426]	Source image containing the angles (Same element type as ipSrcImageMag)
ipDestImageX	Reference To ITcVnImage [▶ 426]	Destination image containing the x values (Same element type as source images. An appropriate destination image will be created if required.)
ipDestImageY	Reference To ITcVnImage [▶ 426]	Destination image containing the y values (Same element type as source images. An appropriate destination image will be created if required.)
bAngleInDegrees	BOOL	Specifys, if the angles are in degrees or radians
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

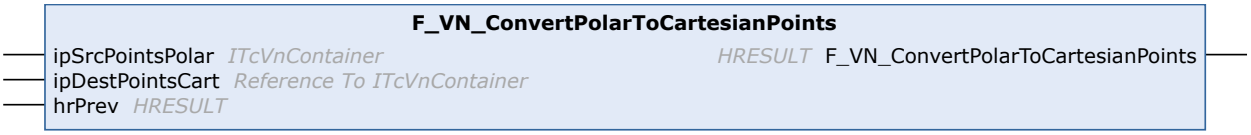
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.24 F_VN_ConvertPolarToCartesianPoints



Converts polar coordinates (magnitude, angle) to cartesian coordinates (x, y).

Syntax

Definition:

```
FUNCTION F_VN_ConvertPolarToCartesianPoints : HRESULT
VAR_INPUT
    ipSrcPointsPolar : ITcVnContainer;
    ipDestPointsCart : Reference To ITcVnContainer;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcPointsPolar	ITcVnContainer [▶ 385]	Container with polar coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestPointsCart	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the converted points in cartesian coordinates (same type ID as ipSrcPointsPolar. Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.25 F_VN_ConvertPolarToCartesianPointsExp

Converts polar coordinates (magnitude, angle) to cartesian coordinates (x, y). (expert function)

Syntax

Definition:

```
FUNCTION F_VN_ConvertPolarToCartesianPointsExp : HRESULT
VAR_INPUT
    ipSrcPointsPolar : ITcVnContainer;
    ipDestPointsCart : Reference To ITcVnContainer;
    bAngleInDegrees  : BOOL;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcPointsPolar	ITcVnContainer [▶ 385]	Container with polar coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestPointsCart	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the converted points in cartesian coordinates (same type ID as ipSrcPointsPolar. Non-zero interface pointers are reused.)
bAngleInDegrees	BOOL	Specifys, if the angles are in degrees or radians
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.26 F_VN_DecomposeAffineTransformation

Decompose an affine transformation matrix and return the rotation angle (clockwise) in radian. The decomposition method assumes the following order of transformation: Translation, Rotation, Scale, and Skew.

Syntax

Definition:

```
FUNCTION F_VN_DecomposeAffineTransformation : HRESULT
VAR_IN_OUT
    aAffineTransform : TcVnMatrix2x3_LREAL;
    fRotationAngle   : LREAL;
END_VAR
VAR_INPUT
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAffineTransform	TcVnMatrix2x3_LREAL [▶ 173]	Affine transformation matrix
fRotationAngle	LREAL	Rotation angle (radian) in the clockwise direction. The rotation center is the origin point (0,0)

Return value

HRESULT [▶ 150]

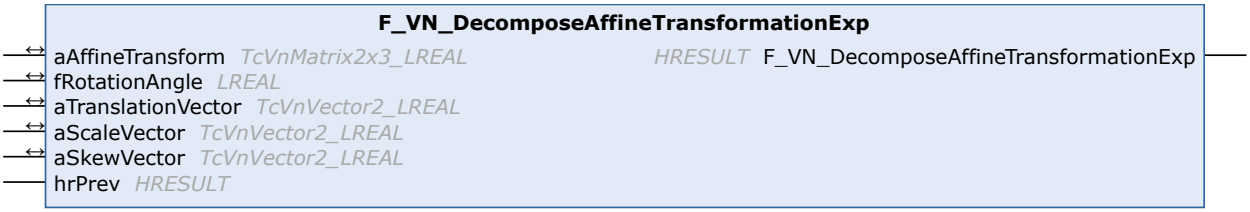
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.27 F_VN_DecomposeAffineTransformationExp



Decompose an affine transformation matrix and return the rotation angle (clockwise) in radian, translation vector (pixels), scale, and skew vectors (angle in radian). The decomposition method assumes the following order of transformation: Translation, Rotation, Scale, and Skew.

Syntax

Definition:

```
FUNCTION F_VN_DecomposeAffineTransformationExp : HRESULT
VAR_IN_OUT
    aAffineTransform   : TcVnMatrix2x3_LREAL;
    fRotationAngle     : LREAL;
    aTranslationVector : TcVnVector2_LREAL;
    aScaleVector       : TcVnVector2_LREAL;
    aSkewVector        : TcVnVector2_LREAL;
END_VAR
VAR_INPUT
    hrPrev             : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAffineTransform	TcVnMatrix2x3_LREAL [▶ 173]	Affine transformation matrix
fRotationAngle	LREAL	Rotation angle (radian) in the clockwise direction. The rotation center is the origin point (0,0)
aTranslationVector	TcVnVector2_LREAL [▶ 173]	2D translation vector. The values are in pixels
aScaleVector	TcVnVector2_LREAL [▶ 173]	2D scale vector
aSkewVector	TcVnVector2_LREAL [▶ 173]	2D skew vector. The values are the skew angles in x and y directions in radian

Return value

HRESULT [▶ 150]

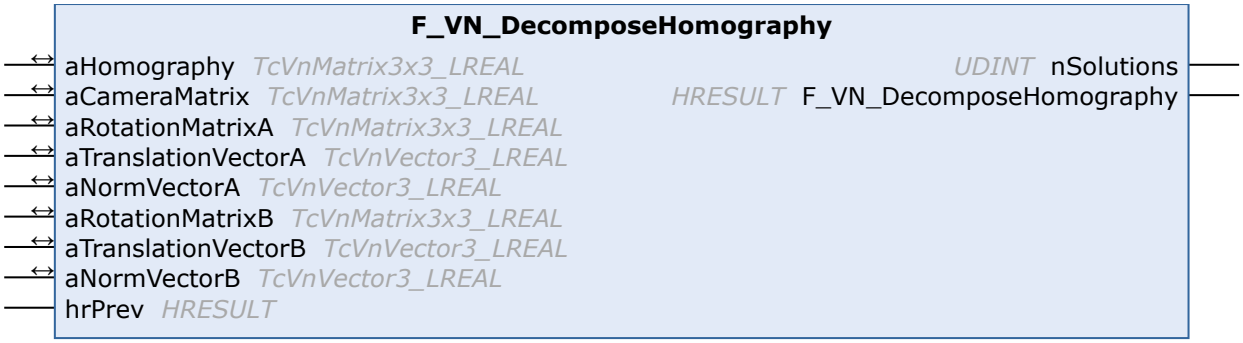
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.28 F_VN_DecomposeHomography



Decompose a homography matrix into four solutions of rotation, translation, and plane normal. It returns two solutions (Ra, Ta, Nb) and (Rb, Tb, Nb). The third and the fourth solutions can be calculated as follows: (Ra, -Ta, -Na) and (Rb, -Tb, -Nb).

Syntax

Definition:

```
FUNCTION F_VN_DecomposeHomography : HRESULT
VAR_IN_OUT
    aHomography         : TcVnMatrix3x3_LREAL;
    aCameraMatrix       : TcVnMatrix3x3_LREAL;
    aRotationMatrixA    : TcVnMatrix3x3_LREAL;
    aTranslationVectorA : TcVnVector3_LREAL;
    aNormVectorA        : TcVnVector3_LREAL;
    aRotationMatrixB    : TcVnMatrix3x3_LREAL;
    aTranslationVectorB : TcVnVector3_LREAL;
    aNormVectorB        : TcVnVector3_LREAL;
END_VAR
```

```
VAR_INPUT
    hrPrev              : HRESULT;
END_VAR
VAR_OUTPUT
    nSolutions          : UDINT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aHomography	TcVnMatrix3x3_LREAL [▶ 173]	Homography matrix (a perspective transformation between two planes)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Camera matrix
aRotationMatrixA	TcVnMatrix3x3_LREAL [▶ 173]	Rotation matrix of the first solution (Ra)
aTranslationVectorA	TcVnVector3_LREAL [▶ 173]	Translation vector of the first solution (Ta)
aNormVectorA	TcVnVector3_LREAL [▶ 173]	Norm vector of the first solution (Na)
aRotationMatrixB	TcVnMatrix3x3_LREAL [▶ 173]	Rotation matrix of the second solution (Rb)
aTranslationVectorB	TcVnVector3_LREAL [▶ 173]	Translation vector of the second solution (Tb)
aNormVectorB	TcVnVector3_LREAL [▶ 173]	Norm vector of the second solution Nb

Outputs

Name	Type	Description
nSolutions	UDINT	Returns the number of solutions. In normal cases, it is equal four.

Return value

HRESULT [▶ 150]

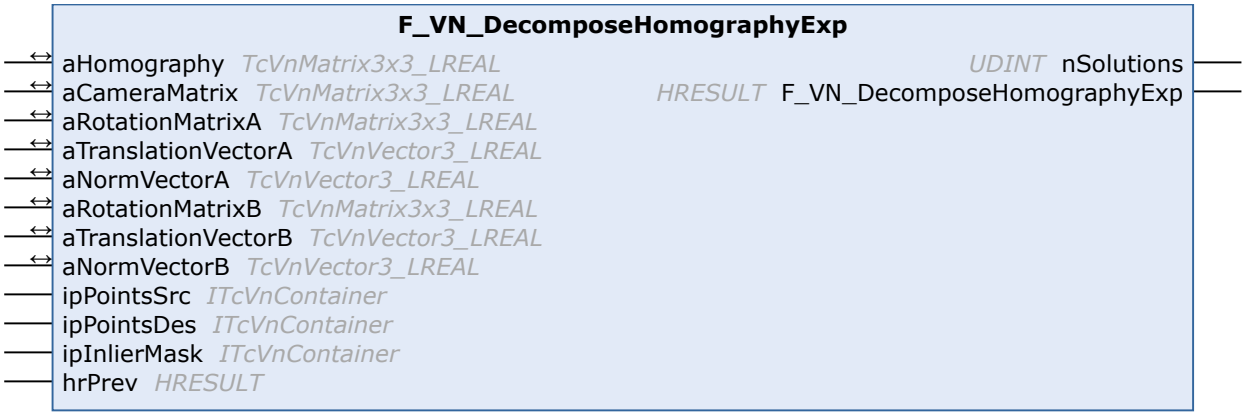
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.29 F_VN_DecomposeHomographyExp



Decompose a homography matrix and verify four solutions of rotation, translation, and plane normal. It uses visible reference points being in front of the camera to confirm which solution(s) of the four solutions (maximum two solutions) are consistent with all reference points.

Syntax

Definition:

```
FUNCTION F_VN_DecomposeHomographyExp : HRESULT
VAR_IN_OUT
    aHomography         : TcVnMatrix3x3_LREAL;
    aCameraMatrix       : TcVnMatrix3x3_LREAL;
    aRotationMatrixA    : TcVnMatrix3x3_LREAL;
    aTranslationVectorA : TcVnVector3_LREAL;
    aNormVectorA        : TcVnVector3_LREAL;
    aRotationMatrixB    : TcVnMatrix3x3_LREAL;
    aTranslationVectorB : TcVnVector3_LREAL;
    aNormVectorB        : TcVnVector3_LREAL;
END_VAR
VAR_INPUT
    ipPointsSrc         : ITcVnContainer;
    ipPointsDes         : ITcVnContainer;
    ipInlierMask        : ITcVnContainer;
    hrPrev              : HRESULT;
END_VAR
VAR_OUTPUT
    nSolutions          : UDINT;
END_VAR
```

Inputs

Name	Type	Description
ipPointsSrc	ITcVnContainer [▶ 385]	Container with source points (ContainerType_Vector_TcVnPoint2_REAL)
ipPointsDes	ITcVnContainer [▶ 385]	Container with destination points (same number as ipPoints1, ContainerType_Vector_TcVnPoint2_REAL)
ipInlierMask	ITcVnContainer [▶ 385]	A mask marking the inliers (optional, set to 0 if not available; ContainerType_Vector_SINT or ContainerType_Vector_USINT)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aHomography	TcVnMatrix3x3_LREAL [▶ 173]	Homography matrix (a perspective transformation between two planes)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Camera matrix
aRotationMatrixA	TcVnMatrix3x3_LREAL [▶ 173]	Rotation matrix of the first solution (Ra)
aTranslationVectorA	TcVnVector3_LREAL [▶ 173]	Translation vector of the first solution (Ta)
aNormVectorA	TcVnVector3_LREAL [▶ 173]	Norm vector of the first solution (Na)
aRotationMatrixB	TcVnMatrix3x3_LREAL [▶ 173]	Rotation matrix of the second solution (Rb)
aTranslationVectorB	TcVnVector3_LREAL [▶ 173]	Translation vector of the second solution (Tb)
aNormVectorB	TcVnVector3_LREAL [▶ 173]	Norm vector of the second solution (Nb)

Outputs

Name	Type	Description
nSolutions	UDINT	Return the number of true potential solutions [0,1, or 2]. 0: no true potential solutions for the given corresponding points are confirmed. 1: the solution (Ra, Ta, Na) is confirmed. 2: both solution (Ra, Ta, Na) and (Rb, Tb, Nb) are confirmed. In all cases the function returns two solutions (Ra, Ta, Nb) and (Rb, Tb, Nb). The third and the fourth solutions can be calculated as follows: (Ra, -Ta, -Na) and (Rb, -Tb, -Nb).

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.30 F_VN_FlipImage

Flip an image (mirror and shift back to prior coordinates).

Syntax

Definition:

```
FUNCTION F_VN_FlipImage : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    eFlipAxis   : ETcVnFlipAxis;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
eFlipAxis	ETcVnFlipAxis [▶ 214]	Selects the axis around which to flip the image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

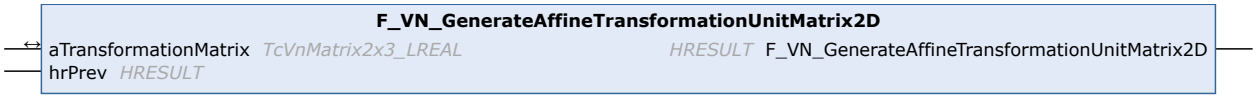
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.31 F_VN_GenerateAffineTransformationUnitMatrix2D



Generate an affine transformation 2D unit matrix.

Syntax

Definition:

```
FUNCTION F_VN_GenerateAffineTransformationUnitMatrix2D : HRESULT
VAR_IN_OUT
    aTransformationMatrix : TcVnMatrix2x3_LREAL;
END_VAR
VAR_INPUT
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 173]	Resulting affine transformation unit matrix

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.32 F_VN_GetAffineTransformation

Calculate the affine transformation between three corresponding point pairs. The points mark the corners of the corresponding triangles.

Syntax

Definition:

```
FUNCTION F_VN_GetAffineTransformation : HRESULT
VAR_IN_OUT
    aSrcPoints            : TcVnArray3_Point2_REAL;
    aDestPoints           : TcVnArray3_Point2_REAL;
    aTransformationMatrix : TcVnMatrix2x3_LREAL;
END_VAR
VAR_INPUT
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aSrcPoints	TcVnArray3_Point2_REAL [▶ 173]	Source points
aDestPoints	TcVnArray3_Point2_REAL [▶ 173]	Destination points
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 173]	Resulting transformation matrix

Return value

HRESULT [▶ 150]

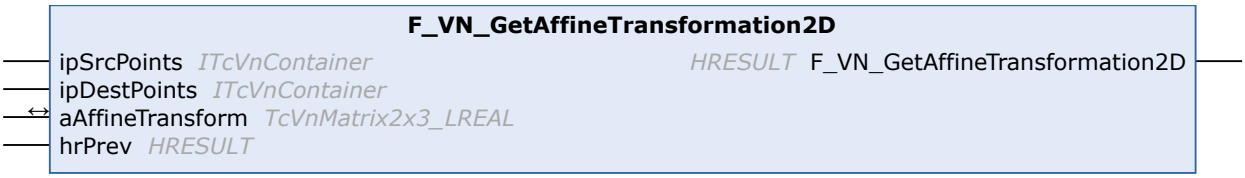
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.33 F_VN_GetAffineTransformation2D



Estimate the 2D affine transformation between two planar point sets using RANSAC algorithm.

Syntax

Definition:

```
FUNCTION F_VN_GetAffineTransformation2D : HRESULT
VAR_INPUT
    ipSrcPoints      : ITcVnContainer;
    ipDestPoints     : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aAffineTransform : TcVnMatrix2x3_LREAL;
END_VAR
VAR_INPUT
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcPoints	ITcVnContainer [▶ 385]	Container with at least 3 source points (ContainerType_Vector_TcVnPoint2_REAL)
ipDestPoints	ITcVnContainer [▶ 385]	Container with destination points (same number as ipSrcPoints, ContainerType_Vector_TcVnPoint2_REAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAffineTransform	TcVnMatrix2x3_LREAL [▶ 173]	Returns the affine transformation matrix, which transforms the source points to the destination points

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.34 F_VN_GetAffineTransformation2DExp

Estimate the affine transformation between two planar point sets. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_GetAffineTransformation2DExp : HRESULT
VAR_INPUT
    ipSrcPoints      : ITcVnContainer;
    ipDestPoints     : ITcVnContainer;
END_VAR
VAR_IN_OUT
    aAffineTransform : TcVnMatrix2x3_LREAL;
END_VAR
VAR_INPUT
    eAlgorithm       : ETcVnEstimationAlgorithm;
    fReprojThreshold : LREAL;
    ipInlierMask     : Reference To ITcVnContainer;
    nMaxIterations   : UDINT;
    fConfidence      : LREAL;
    nRefineIters     : UDINT;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcPoints	ITcVnContainer [▶ 385]	Container with at least 3 source points (ContainerType_Vector_TcVnPoint2_REAL)
ipDestPoints	ITcVnContainer [▶ 385]	Container with destination points (same number as ipSrcPoints, ContainerType_Vector_TcVnPoint2_REAL)
eAlgorithm	ETcVnEstimationAlgorithm [▶ 211]	Estimation algorithm (only RANSAC and LMEDS are supported)
fReprojThreshold	LREAL	Maximum allowed reprojection error to treat a point pair as an inlier
ipInlierMask	Reference To ITcVnContainer [▶ 385]	Returns a mask marking the inliers (optional, set to 0 if not required; ContainerType_Vector_USINT)
nMaxIterations	UDINT	Maximum number of iterations
fConfidence	LREAL	Confidence (0..1)
nRefineIters	UDINT	Maximum number of iterations of Levenberg-Marquardt algorithm to refine further the affine transform (using only inliers). Set to 0 to disable refining.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAffineTransform	TcVnMatrix2x3_LREAL [▶ 173]	Returns the affine transformation matrix, which transforms the source points to the destination points

Return value

HRESULT [▶ 150]

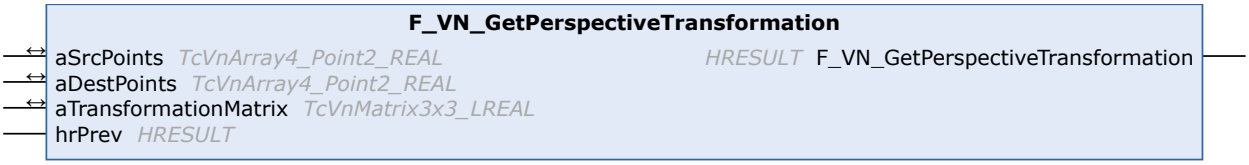
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.35 F_VN_GetPerspectiveTransformation



Calculate the perspective transformation between four corresponding point pairs. The points mark the corners of the corresponding rectangles.

Syntax

Definition:

```
FUNCTION F_VN_GetPerspectiveTransformation : HRESULT
VAR_IN_OUT
    aSrcPoints            : TcVnArray4_Point2_REAL;
    aDestPoints           : TcVnArray4_Point2_REAL;
    aTransformationMatrix : TcVnMatrix3x3_LREAL;
END_VAR
VAR_INPUT
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aSrcPoints	TcVnArray4_Point2_REAL [▶ 173]	Source points
aDestPoints	TcVnArray4_Point2_REAL [▶ 173]	Destination points
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Resulting transformation matrix

Return value

HRESULT [▶ 150]

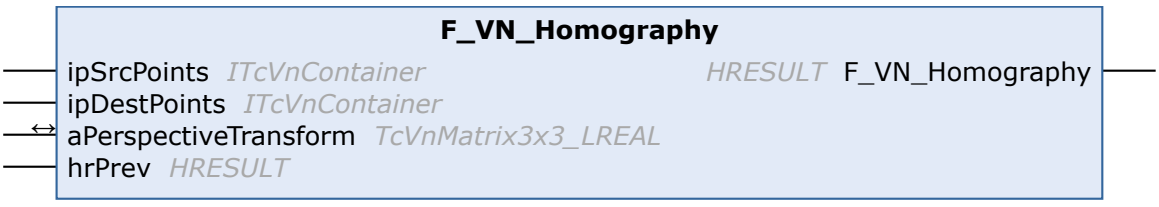
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.36 F_VN_Homography



Find the homography (perspective transformation) between two planar point sets using RANSAC algorithm.

Syntax

Definition:

```
FUNCTION F_VN_Homography : HRESULT
VAR_INPUT
    ipSrcPoints           : ITcVnContainer;
    ipDestPoints          : ITcVnContainer;
END_VAR
VAR_IN_OUT
```

```
    aPerspectiveTransform : TcVnMatrix3x3_LREAL;
END_VAR
VAR_INPUT
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcPoints	ITcVnContainer [▶ 385]	Container with at least 4 source points (ContainerType_Vector_TcVnPoint2_REAL)
ipDestPoints	ITcVnContainer [▶ 385]	Container with destination points (same number as ipSrcPoints, ContainerType_Vector_TcVnPoint2_REAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 173]	Returns the perspective transformation matrix, which transforms the source points to the destination points

Return value

HRESULT [▶ 150]

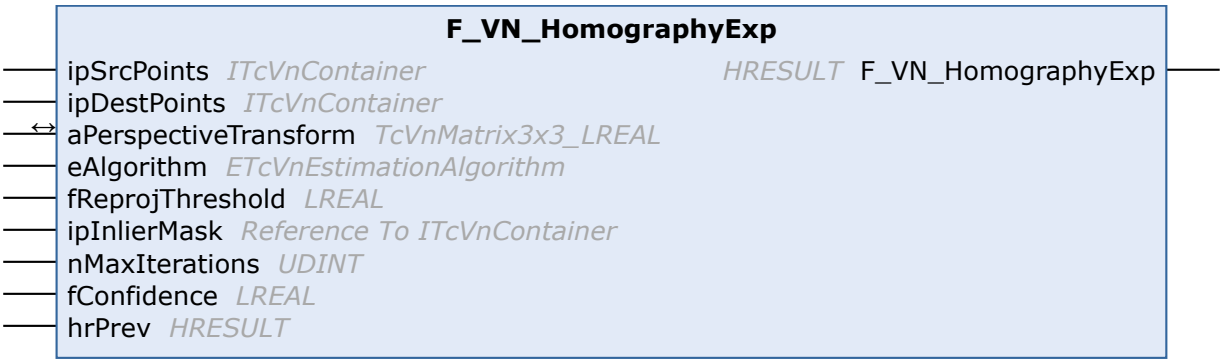
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.37 F_VN_HomographyExp



Find the homography (perspective transformation) between two planar point sets. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_HomographyExp : HRESULT
VAR_INPUT
    ipSrcPoints           : ITcVnContainer;
    ipDestPoints          : ITcVnContainer;
```

```
END_VAR
VAR_IN_OUT
    aPerspectiveTransform : TcVnMatrix3x3_LREAL;
END_VAR
VAR_INPUT
    eAlgorithm            : ETcVnEstimationAlgorithm;
    fReprojThreshold      : LREAL;
    ipInlierMask          : Reference To ITcVnContainer;
    nMaxIterations        : UDINT;
    fConfidence           : LREAL;
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcPoints	ITcVnContainer [▶ 385]	Container with at least 4 source points (ContainerType_Vector_TcVnPoint2_REAL)
ipDestPoints	ITcVnContainer [▶ 385]	Container with destination points (same number as ipSrcPoints, ContainerType_Vector_TcVnPoint2_REAL)
eAlgorithm	ETcVnEstimationAlgorithm [▶ 211]	Estimation algorithm
fReprojThreshold	LREAL	Maximum allowed reprojection error to treat a point pair as an inlier (only for RANSAC, RHO)
ipInlierMask	Reference To ITcVnContainer [▶ 385]	Returns a mask marking the inliers (optional, set to 0 if not required; ContainerType_Vector_SINT; only for RANSAC, LMEDS)
nMaxIterations	UDINT	Maximum number of RANSAC iterations
fConfidence	LREAL	Confidence (0..1)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 173]	Returns the perspective transformation matrix, which transforms the source points to the destination points

Return value

HRESULT [▶ 150]

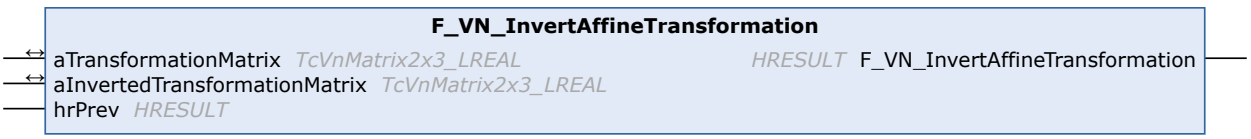
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.38 F_VN_InvertAffineTransformation



Invert a 2D affine transformation matrix. The inverted transformation compensates the affine transformation on an image if it is used in the F_VN_WarpAffine function.

Syntax

Definition:

```
FUNCTION F_VN_InvertAffineTransformation : HRESULT
VAR_IN_OUT
    aTransformationMatrix         : TcVnMatrix2x3_LREAL;
    aInvertedTransformationMatrix : TcVnMatrix2x3_LREAL;
END_VAR
VAR_INPUT
    hrPrev                        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 173]	Source affine transformation matrix
aInvertedTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 173]	Resulting inverted affine transformation matrix

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.39 F_VN_PerspectiveTransformation

Apply a perspective transformation (rotation + translation, e.g. from extrinsic calibration) to 3D point coordinates.

Syntax

Definition:

```
FUNCTION F_VN_PerspectiveTransformation : HRESULT
VAR_INPUT
    ipSrcPoints        : ITcVnContainer;
    ipDestPoints       : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    aRotationMatrix    : TcVnMatrix3x3_LREAL;
    aTranslationVector : TcVnVector3_LREAL;
END_VAR
VAR_INPUT
    hrPrev             : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcPoints	ITcVnContainer [▶ 385]	Container with 3D source points (TcVnPoint3_REAL or TcVnPoint3_LREAL)
ipDestPoints	Reference To ITcVnContainer [▶ 385]	Returns the transformed points (same type as ipSrcPoints)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 173]	Translation vector

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.40 F_VN_PyramidDown

Downsamples an image to half width and height.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_PyramidDown : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.41 F_VN_PyramidUp

Upsamples an image to double width and height.

Syntax

Definition:

```
FUNCTION F_VN_PyramidUp : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.42 F_VN_RemapImageToLogPolarSpace

Remap an image to log-polar space.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_RemapImageToLogPolarSpace : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    aCenter     : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (Must not be the same as ipSrcImage! An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCenter	TcVnPoint2_REAL [▶ 171]	Center point for the transformation

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.43 F_VN_RemapImageToLogPolarSpaceExp

Remap an image to log-polar space. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_RemapImageToLogPolarSpaceExp : HRESULT
VAR_INPUT
    ipSrcImage          : ITcVnImage;
    ipDestImage         : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    aCenter             : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    fScale              : LREAL;
    eInterpolationType  : ETcVnInterpolationType;
    hrPrev              : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (Must not be the same as ipSrcImage! An appropriate destination image will be created if required.)
fScale	LREAL	Magnitude scale parameter, i.e. image width / ln(radius) (set to 0 for auto select)
eInterpolationType	ETcVnInterpolationType [▶ 217]	Interpolation type
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCenter	TcVnPoint2_REAL [▶ 171]	Center point for the transformation

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.44 F_VN_RemapImageToLogPolarSpaceExp2

Remap an image to log-polar space. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_RemapImageToLogPolarSpaceExp2 : HRESULT
VAR_INPUT
    ipSrcImage          : ITcVnImage;
    ipDestImage         : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    aCenter             : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    fScale              : LREAL;
    eInterpolationType  : ETcVnInterpolationType;
    nDestWidth          : DINT;
    nDestHeight         : DINT;
    hrPrev              : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (Must not be the same as ipSrcImage! An appropriate destination image will be created if required.)
fScale	LREAL	Magnitude scale parameter, i.e. image width / ln(radius) (set to 0 for auto select)
eInterpolationType	ETcVnInterpolationType [▶ 217]	Interpolation type
nDestWidth	DINT	Destination image width (set to -1 for source image width, 0 to auto scale to transformed radius, > 0 for a user defined width)
nDestHeight	DINT	destination image height (set to -1 for source image height, 0 to auto scale to transformed radius * PI, > 0 for a user defined height)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCenter	TcVnPoint2_REAL [▶ 171]	Center point for the transformation

Return value

HRESULT [▶ 150]

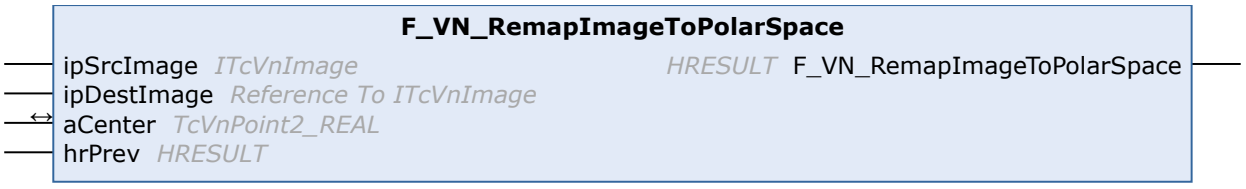
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.45 F_VN_RemapImageToPolarSpace



Remap an image to polar space.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_RemapImageToPolarSpace : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
```

```
    aCenter     : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (Must not be the same as ipSrcImage! An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCenter	TcVnPoint2_REAL [▶ 171]	Center point for the transformation

Return value

HRESULT [▶ 150]

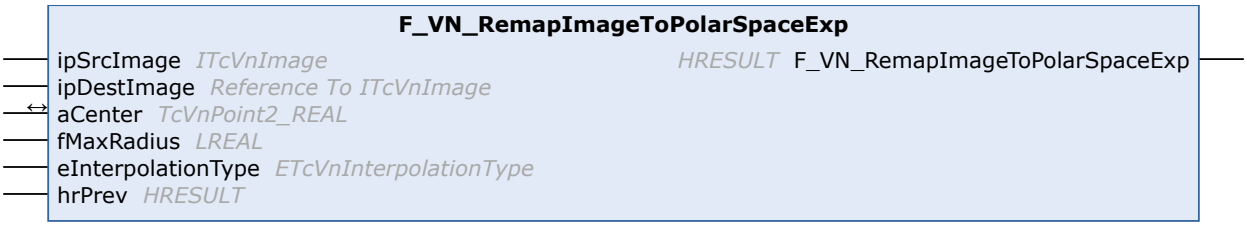
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.46 F_VN_RemapImageToPolarSpaceExp



Remap an image to polar space. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_RemapImageToPolarSpaceExp : HRESULT
VAR_INPUT
    ipSrcImage          : ITcVnImage;
    ipDestImage         : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    aCenter             : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    fMaxRadius          : LREAL;
```

```
    eInterpolationType : ETcVnInterpolationType;
    hrPrev             : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (Must not be the same as ipSrcImage! An appropriate destination image will be created if required.)
fMaxRadius	LREAL	Maximum radius for the transformation (set to 0 for auto select)
eInterpolationType	ETcVnInterpolationType [▶ 217]	Interpolation type
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCenter	TcVnPoint2_REAL [▶ 171]	Center point for the transformation

Return value

HRESULT [▶ 150]

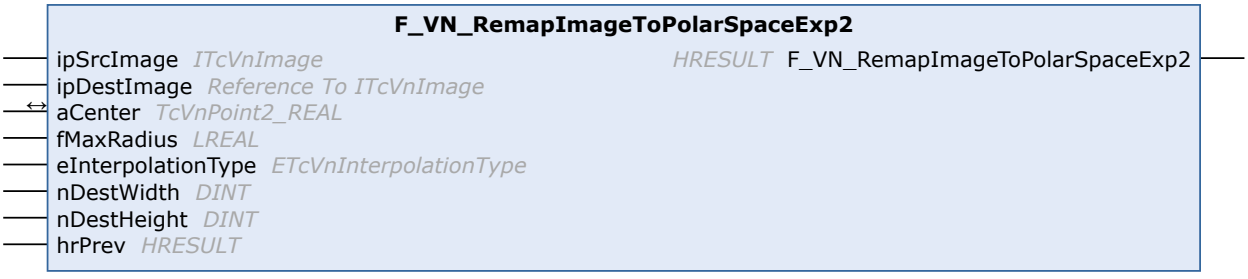
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.47 F_VN_RemapImageToPolarSpaceExp2



Remap an image to polar space. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_RemapImageToPolarSpaceExp2 : HRESULT
VAR_INPUT
    ipSrcImage         : ITcVnImage;
    ipDestImage        : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
```

```
    aCenter            : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    fMaxRadius         : LREAL;
    eInterpolationType : ETcVnInterpolationType;
    nDestWidth         : DINT;
    nDestHeight        : DINT;
    hrPrev             : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (Must not be the same as ipSrcImage! An appropriate destination image will be created if required.)
fMaxRadius	LREAL	Maximum radius for the transformation (set to 0 for auto select)
eInterpolationType	ETcVnInterpolationType [▶ 217]	Interpolation type
nDestWidth	DINT	Destination image width (set to -1 for source image width, 0 to auto scale to transformed radius, > 0 for a user defined width)
nDestHeight	DINT	destination image height (set to -1 for source image height, 0 to auto scale to transformed radius * PI, > 0 for a user defined height)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCenter	TcVnPoint2_REAL [▶ 171]	Center point for the transformation

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.48 F_VN_ResizeImage

Resize an image using a specific interpolation type.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_ResizeImage : HRESULT
VAR_INPUT
    ipSrcImage          : ITcVnImage;
    ipDestImage         : Reference To ITcVnImage;
    nWidth              : UDINT;
    nHeight             : UDINT;
    eInterpolationType  : ETcVnInterpolationType;
    hrPrev              : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate image will be created if required.)
nWidth	UDINT	New width
nHeight	UDINT	New height
eInterpolationType	ETcVnInterpolationType [▶ 217]	Interpolation type
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

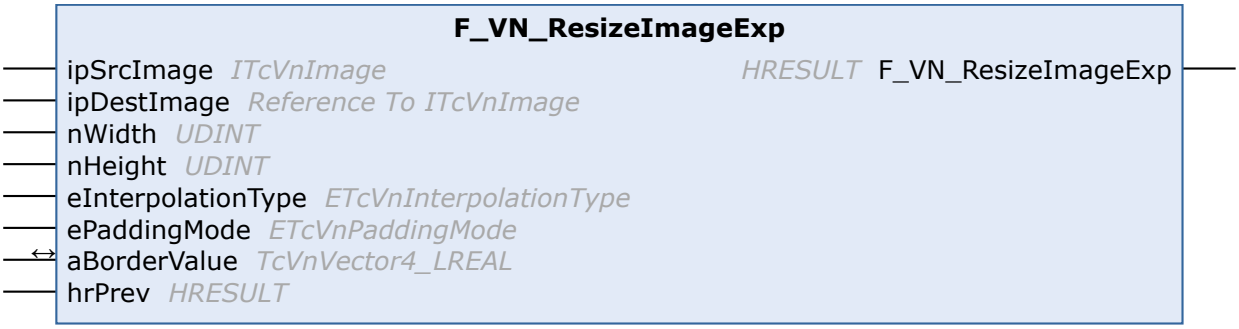
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.49 F_VN_ResizeImageExp



Resize an image using a specific interpolation type while maintaining its aspect ratio. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_ResizeImageExp : HRESULT
VAR_INPUT
    ipSrcImage          : ITcVnImage;
    ipDestImage         : Reference To ITcVnImage;
    nWidth              : UDINT;
    nHeight             : UDINT;
    eInterpolationType  : ETcVnInterpolationType;
    ePaddingMode        : ETcVnPaddingMode;
END_VAR
VAR_IN_OUT
    aBorderValue        : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev              : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate image will be created if required.)
nWidth	UDINT	New width
nHeight	UDINT	New height
eInterpolationType	ETcVnInterpolationType [▶ 217]	Interpolation type
ePaddingMode	ETcVnPaddingMode [▶ 224]	Image padding mode
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aBorderValue	TcVnVector4_LREAL [▶ 173]	Border value, if TCVN_PM_LETTERBOX is used

Return value

HRESULT [▶ 150]

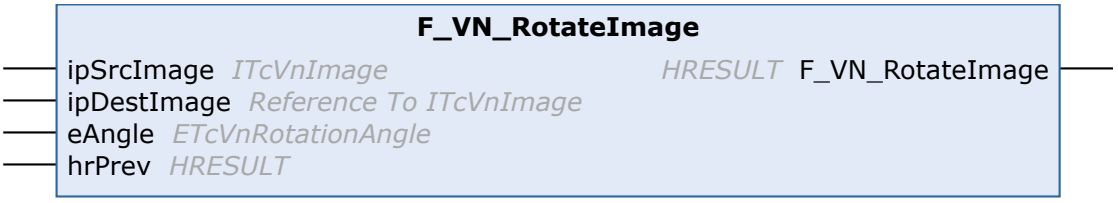
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.50 F_VN_RotateImage



Rotate an image by 90, 180, or 270 degrees in clockwise direction.

Syntax

Definition:

```
FUNCTION F_VN_RotateImage : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    eAngle      : ETcVnRotationAngle;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate image will be created if required.)
eAngle	ETcVnRotationAngle [▶ 227]	Rotation angle
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

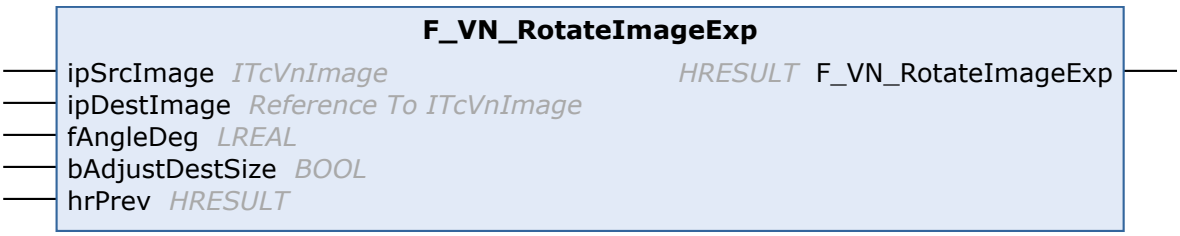
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.51 F_VN_RotateImageExp



Rotate an image by an individual rotation angle in degrees.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_RotateImageExp : HRESULT
VAR_INPUT
    ipSrcImage      : ITcVnImage;
    ipDestImage     : Reference To ITcVnImage;
    fAngleDeg       : LREAL;
    bAdjustDestSize : BOOL;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate image will be created if required.)
fAngleDeg	LREAL	Rotation angle in degree (positive: counter-clockwise, negative: clockwise)
bAdjustDestSize	BOOL	If true, ipDestImage size is adjusted so that the whole rotated ipSrcImage is contained. Otherwise, ipDestImage size is set to ipSrcImage size.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

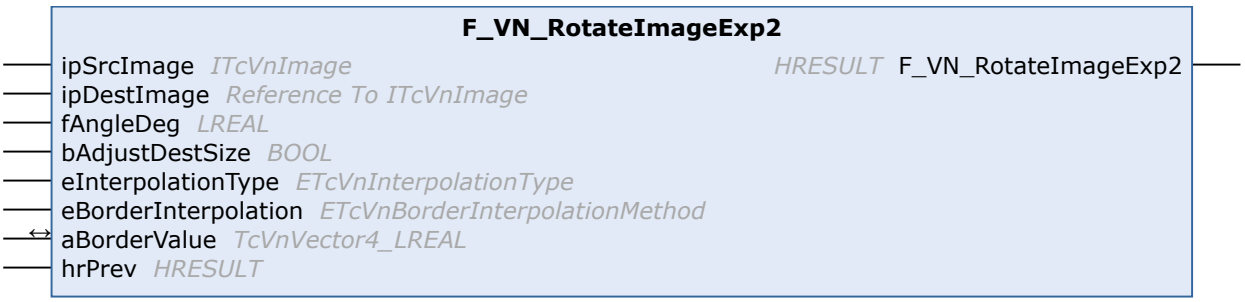
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.52 F_VN_RotateImageExp2



Rotate an image by an individual rotation angle in degrees. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_RotateImageExp2 : HRESULT
VAR_INPUT
    ipSrcImage           : ITcVnImage;
    ipDestImage          : Reference To ITcVnImage;
    fAngleDeg            : LREAL;
    bAdjustDestSize      : BOOL;
    eInterpolationType   : ETcVnInterpolationType;
    eBorderInterpolation : ETcVnBorderInterpolationMethod;
END_VAR
VAR_IN_OUT
    aBorderValue         : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev               : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate image will be created if required.)
fAngleDeg	LREAL	Rotation angle in degree (positive: counter-clockwise, negative: clockwise)
bAdjustDestSize	BOOL	If true, ipDestImage size is adjusted so that the whole rotated ipSrcImage is contained. Otherwise, ipDestImage size is set to ipSrcImage size.
eInterpolationType	ETcVnInterpolationType [▶ 217]	Interpolation method
eBorderInterpolation	ETcVnBorderInterpolationMethod [▶ 177]	Border interpolation method (ISOLATED not supported)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aBorderValue	TcVnVector4_LREAL [▶ 173]	Border value, if CONSTANT is used

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.53 F_VN_WarpAffine

Apply an affine transformation to an image.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_WarpAffine : HRESULT
VAR_INPUT
    ipSrcImage            : ITcVnImage;
    ipDestImage           : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    aTransformationMatrix : TcVnMatrix2x3_LREAL;
END_VAR
VAR_INPUT
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 173]	Affine transformation matrix

Return value

HRESULT [▶ 150]

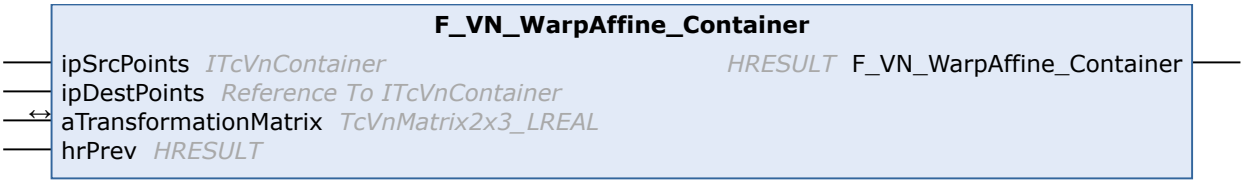
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.54 F_VN_WarpAffine_Container



Apply an affine transformation to a container of 2D points.

Syntax

Definition:

```
FUNCTION F_VN_WarpAffine_Container : HRESULT
VAR_INPUT
    ipSrcPoints           : ITcVnContainer;
    ipDestPoints          : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    aTransformationMatrix : TcVnMatrix2x3_LREAL;
END_VAR
VAR_INPUT
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcPoints	ITcVnContainer [▶ 385]	Source points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestPoints	Reference To ITcVnContainer [▶ 385]	Returns the transformed points (same type ID as ipSrcPoints; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 173]	Affine transformation matrix

Return value

HRESULT [▶ 150]

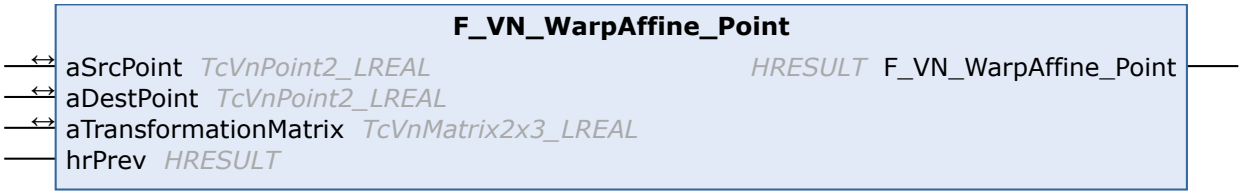
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.55 F_VN_WarpAffine_Point



Apply an affine transformation to a 2D point.

Syntax

Definition:

```
FUNCTION F_VN_WarpAffine_Point : HRESULT
VAR_IN_OUT
    aSrcPoint             : TcVnPoint2_LREAL;
    aDestPoint            : TcVnPoint2_LREAL;
    aTransformationMatrix : TcVnMatrix2x3_LREAL;
END_VAR
VAR_INPUT
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aSrcPoint	TcVnPoint2_LREAL [▶ 171]	Source point
aDestPoint	TcVnPoint2_LREAL [▶ 171]	Destination point
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 173]	Affine transformation matrix

Return value

HRESULT [▶ 150]

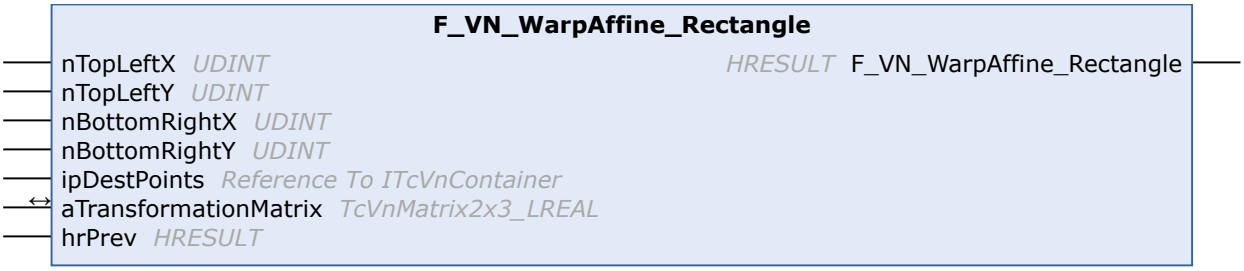
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.56 F_VN_WarpAffine_Rectangle



Use an affine transform to warp a rectangle.

Syntax

Definition:

```
FUNCTION F_VN_WarpAffine_Rectangle : HRESULT
VAR_INPUT
    nTopLeftX             : UDINT;
    nTopLeftY             : UDINT;
    nBottomRightX         : UDINT;
    nBottomRightY         : UDINT;
    ipDestPoints          : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    aTransformationMatrix : TcVnMatrix2x3_LREAL;
END_VAR
VAR_INPUT
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nTopLeftX	UDINT	x coordinate of the top left corner
nTopLeftY	UDINT	y coordinate of the top left corner
nBottomRightX	UDINT	x coordinate of the bottom right corner
nBottomRightY	UDINT	y coordinate of the bottom right corner
ipDestPoints	Reference To ITcVnContainer [▶ 385]	Returns the 4 transformed points (ContainerType_Vector_TcVnPoint2_REAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 173]	Affine transformation matrix

Return value

HRESULT [▶ 150]

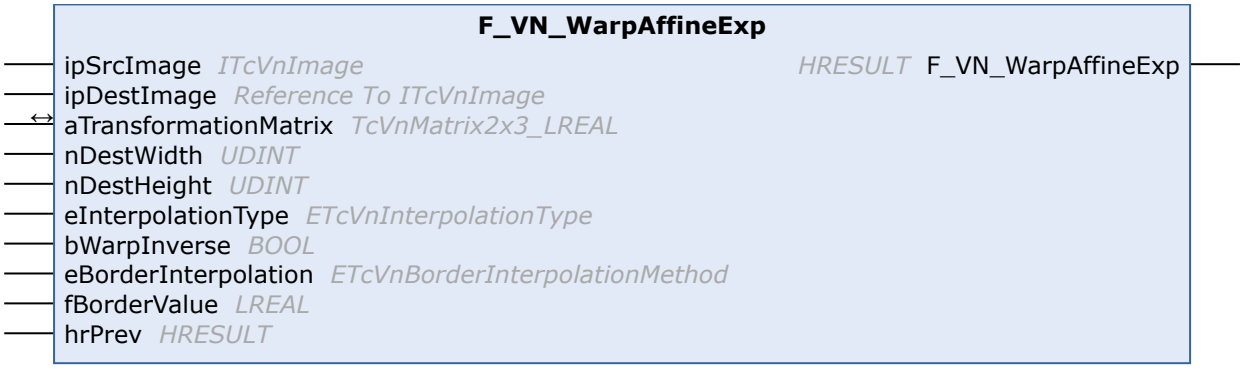
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.57 F_VN_WarpAffineExp



Apply an affine transformation to an image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_WarpAffineExp : HRESULT
VAR_INPUT
    ipSrcImage            : ITcVnImage;
    ipDestImage           : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    aTransformationMatrix : TcVnMatrix2x3_LREAL;
END_VAR
VAR_INPUT
    nDestWidth            : UDINT;
    nDestHeight           : UDINT;
    eInterpolationType    : ETcVnInterpolationType;
    bWarpInverse          : BOOL;
    eBorderInterpolation  : ETcVnBorderInterpolationMethod;
    fBorderValue          : LREAL;
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate image will be created if required.)
nDestWidth	UDINT	Width of the destination image
nDestHeight	UDINT	Height of the destination image
eInterpolationType	ETcVnInterpolationType [▶ 217]	Interpolation method
bWarpInverse	BOOL	Handle aTransformationMatrix as the inverse transformation
eBorderInterpolation	ETcVnBorderInterpolationMethod [▶ 177]	Border interpolation method (ISOLATED not supported)
fBorderValue	LREAL	Border value, if CONSTANT is used
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 173]	Affine transformation matrix

Return value

HRESULT [▶ 150]

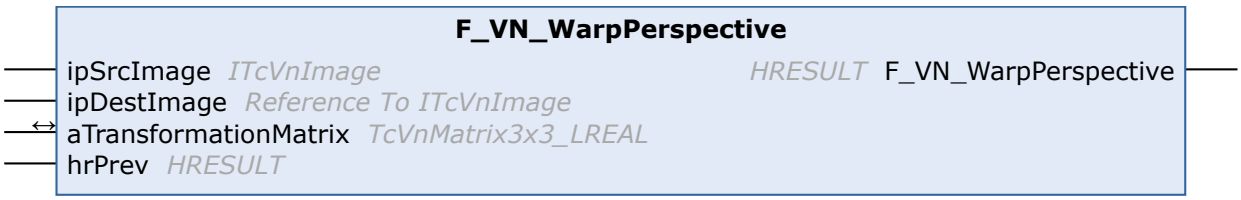
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.58 F_VN_WarpPerspective



Apply a perspective transformation to an image.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_WarpPerspective : HRESULT
VAR_INPUT
    ipSrcImage            : ITcVnImage;
    ipDestImage           : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    aTransformationMatrix : TcVnMatrix3x3_LREAL;
END_VAR
VAR_INPUT
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Perspective transformation matrix

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.59 F_VN_WarpPerspective_Container

Apply a perspective transformation to a container of 2D points.

Syntax

Definition:

```
FUNCTION F_VN_WarpPerspective_Container : HRESULT
VAR_INPUT
    ipSrcPoints           : ITcVnContainer;
    ipDestPoints          : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    aTransformationMatrix : TcVnMatrix3x3_LREAL;
```

```
END_VAR
VAR_INPUT
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcPoints	ITcVnContainer [▶ 385]	Source points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestPoints	Reference To ITcVnContainer [▶ 385]	Returns the transformed points (same type ID as ipSrcPoints; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Perspective transformation matrix

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.60 F_VN_WarpPerspective_Point

Apply a perspective transformation to a 2D point.

Syntax

Definition:

```
FUNCTION F_VN_WarpPerspective_Point : HRESULT
VAR_IN_OUT
    aSrcPoint             : TcVnPoint2_LREAL;
    aDestPoint            : TcVnPoint2_LREAL;
    aTransformationMatrix : TcVnMatrix3x3_LREAL;
END_VAR
VAR_INPUT
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aSrcPoint	TcVnPoint2_LREAL [▶ 171]	Source point
aDestPoint	TcVnPoint2_LREAL [▶ 171]	Destination point
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Perspective transformation matrix

Return value

HRESULT [▶ 150]

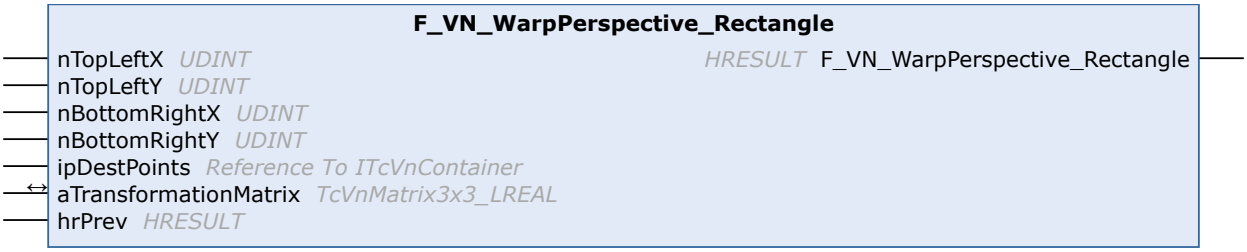
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.61 F_VN_WarpPerspective_Rectangle



Use a perspective transform to warp a rectangle.

Syntax

Definition:

```
FUNCTION F_VN_WarpPerspective_Rectangle : HRESULT
VAR_INPUT
    nTopLeftX             : UDINT;
    nTopLeftY             : UDINT;
    nBottomRightX         : UDINT;
    nBottomRightY         : UDINT;
    ipDestPoints          : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    aTransformationMatrix : TcVnMatrix3x3_LREAL;
END_VAR
VAR_INPUT
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nTopLeftX	UDINT	x coordinate of the top left corner
nTopLeftY	UDINT	y coordinate of the top left corner
nBottomRightX	UDINT	x coordinate of the bottom right corner
nBottomRightY	UDINT	y coordinate of the bottom right corner
ipDestPoints	Reference To ITcVnContainer [▶ 385]	Returns the 4 transformed points (ContainerType_Vector_TcVnPoint2_REAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Perspective transformation matrix

Return value

HRESULT [▶ 150]

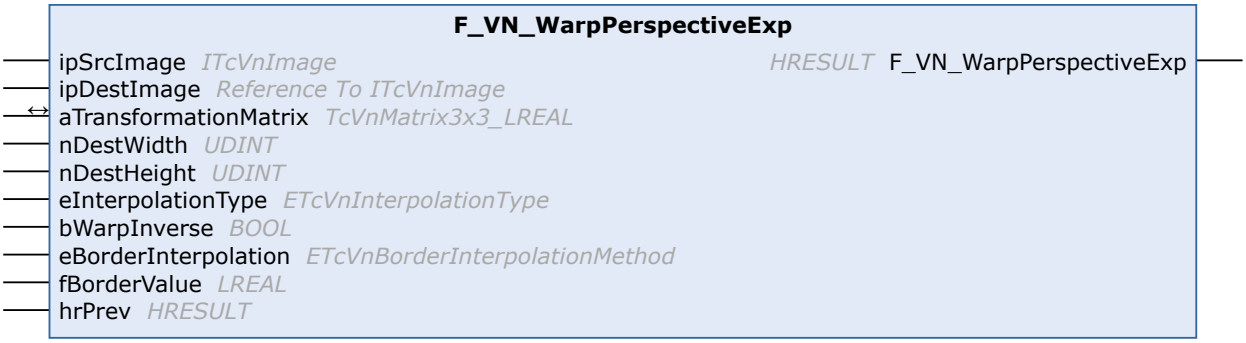
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.13.62 F_VN_WarpPerspectiveExp



Apply a perspective transformation to an image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_WarpPerspectiveExp : HRESULT
VAR_INPUT
    ipSrcImage            : ITcVnImage;
    ipDestImage           : Reference To ITcVnImage;
END_VAR
```

```
VAR_IN_OUT
    aTransformationMatrix : TcVnMatrix3x3_LREAL;
END_VAR
VAR_INPUT
    nDestWidth            : UDINT;
    nDestHeight           : UDINT;
    eInterpolationType    : ETcVnInterpolationType;
    bWarpInverse          : BOOL;
    eBorderInterpolation  : ETcVnBorderInterpolationMethod;
    fBorderValue          : LREAL;
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate image will be created if required.)
nDestWidth	UDINT	Width of the destination image
nDestHeight	UDINT	Height of the destination image
eInterpolationType	ETcVnInterpolationType [▶ 217]	Interpolation method
bWarpInverse	BOOL	Handle aTransformationMatrix as the inverse transformation
eBorderInterpolation	ETcVnBorderInterpolationMethod [▶ 177]	Border interpolation method (ISOLATED not supported)
fBorderValue	LREAL	Border value, if CONSTANT is used
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Perspective transformation matrix

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14 Image Analysis

Diese Gruppe enthält Funktionen zur Extraktion von Informationen aus Bildern.

Funktionen

Objektdetektion [▶ 1205]

- Kontursuche
- Objektfindung durch Hough Transformation
- Matching durch Bildvergleich

Statistische Bildmerkmale

- F_VN_ImageAverage(Exp) [▶ 1247]
- F_VN_ImageAverageStdDev(Exp) [▶ 1248]
- F_VN_ImageAbsoluteAverage [▶ 1262]
- F_VN_ImageMedian(Exp) [▶ 1253]
- F_VN_MaxPixelValue(Exp) [▶ 1256]
- F_VN_MinPixelValue(Exp) [▶ 1258]

Kantendetektion

- F_VN_CannyEdgeDetection(Exp) [▶ 1233]

Finden zusammenhängender Bildregionen

- F_VN_ConnectedComponents(Exp) [▶ 1236]
- F_VN_ConnectedComponentsWithStats (Exp) [▶ 1239]
- F_VN_GetConnectedComponent(Exp) [▶ 1245]

Ausrichtung einer Region

- F_VN_RegionOrientation(Exp) [▶ 1260]
- F_VN_UprightInnerRectangle(Exp) [▶ 1262]

Sonstiges

- F_VN_CountNonZeroPixels [▶ 1241]
- F_VN_DistanceTransformation(Exp) [▶ 1242]
- F_VN_ImageCenterOfMass(Exp) [▶ 1250]
- F_VN_ImageMoments [▶ 1255]

Beispiele

Image Analysis [▶ 2867]

6.1.4.14.1 Object Detection

Diese Untergruppe enthält Funktionen zur Objekterkennung.

Funktionen

Objektdetektion

- F_VN_DetectBlobs(Exp) [▶ 1207]
- F_VN_FindContours(Exp) [▶ 1219]
- F_VN_FindContourHierarchyExp [▶ 1212]
- F_VN_HoughCircles(Exp) [▶ 1222]

- F_VN_HoughLines(Exp) [▶ 1224]
- F_VN_HoughLinesP(Exp) [▶ 1226]

Matching

- F_VN_MatchImageHuMoments [▶ 1227]
- F_VN_MatchTemplate(Exp) [▶ 1232]
- F_VN_MatchTemplateAndEvaluate(Exp) [▶ 1230]

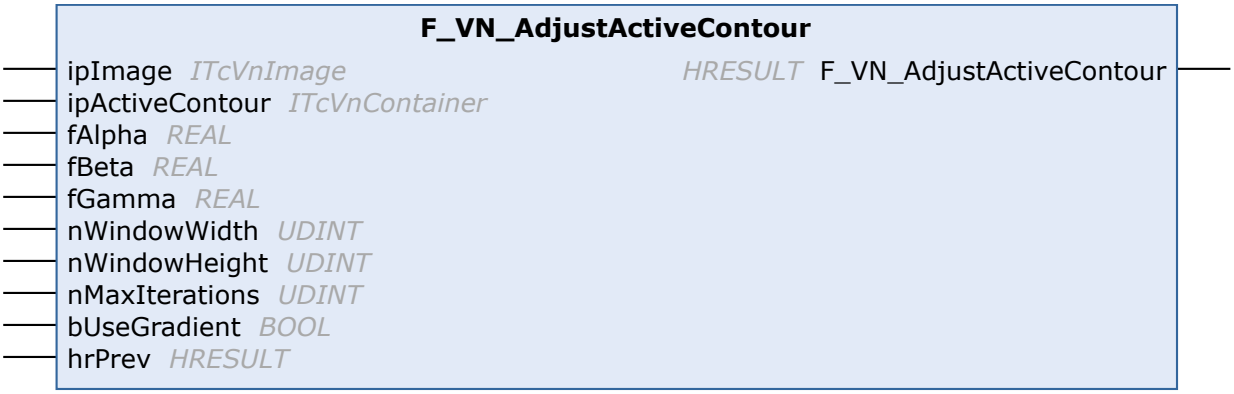
Sonstiges

- F_VN_AdjustActiveContour [▶ 1206]

Beispiele

Object Detection [▶ 2867]

6.1.4.14.1.1 F_VN_AdjustActiveContour



Adjust active contour (snake) in order to minimize its cummulative (internal and external) energy.

Syntax

Definition:

```
FUNCTION F_VN_AdjustActiveContour : HRESULT
VAR_INPUT
    ipImage         : ITcVnImage;
    ipActiveContour : ITcVnContainer;
    fAlpha          : REAL;
    fBeta           : REAL;
    fGamma          : REAL;
    nWindowWidth    : UDINT;
    nWindowHeight   : UDINT;
    nMaxIterations  : UDINT;
    bUseGradient    : BOOL;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Source image (1 channel, USINT elements)
ipActiveContour	ITcVnContainer [▶ 385]	Initial contour, which will be adjusted
fAlpha	REAL	Continuity energy coefficient
fBeta	REAL	Curvature energy coefficient
fGamma	REAL	Image energy coefficient
nWindowWidth	UDINT	Window width (3, 5, 7, ...)
nWindowHeight	UDINT	Window height (3, 5, 7, ...)
nMaxIterations	UDINT	Maximum iterations
bUseGradient	BOOL	If true, the gradient magnitude is used as image energy (otherwise: pixel intensity)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

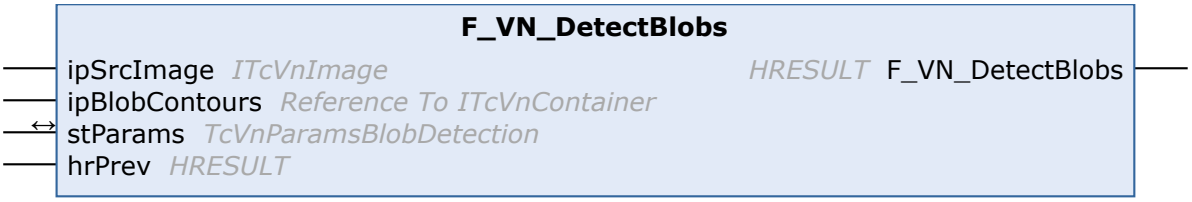
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.1.2 F_VN_DetectBlobs



Detects blob-contours. Applies a threshold, a contour search and offers several options for filtering the found contours. Provides easy setup for multiple thresholds and combination of results.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_DetectBlobs : HRESULT
VAR_INPUT
    ipSrcImage     : ITcVnImage;
    ipBlobContours : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    stParams       : TcVnParamsBlobDetection;
END_VAR
```

```
VAR_INPUT
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT or REAL, 1 channel or 3 channel (3 channel input is expected to be RGB and internally converted to Gray))
ipBlobContours	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the found contours (ContainerType_Vector_Vector_TcVnPoint2_DINT; The elements of this container are single contours. Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsBlobDetection [▶ 247]	Parameters to filter the detected contours.

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_DetectBlobs` detektiert Blob-Konturen anhand von definierbaren Merkmalen im Eingangsbild. Intern geschieht dies durch eine Konturfindung analog zu F_VN_FindContours [▶ 1216] und anschließender Filterung der gefundenen Konturen anhand von Konturmerkmalen.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1- oder 3-kanaliges Bild vom Typ `USINT` oder `REAL` sein. Dabei werden 3-kanalige Bilder als RGB-Bilder interpretiert und intern in ein Grauwertbild konvertiert.

Liste an gefundenen Konturen (Rückgabewert)

Alle im Eingangsbild gefundenen Konturen werden in dem Container `ipBlobContours` zurückgegeben.

Parameter zur Blob-Detektion

Die Parameter `stParams` der Struktur TcVnParamsBlobDetection [▶ 247] legen fest, welche Merkmale eine Kontur im Eingangsbild haben muss, um gefunden und zurückgegeben zu werden.

Expert-Parameter

Die Expert-Variante F_VN_DetectBlobsExp [▶ 1210] enthält zusätzliche Parameter.

Blobs-Parameter

Schwellwert

Im ersten Schritt erfolgt intern eine Kontursuche anhand eines oder mehrerer Binärbilder. Dazu wird das Eingangsbild anhand eines oder mehrerer Schwellwerte, sowie des Schwellwerttyps in die Binärbilder konvertiert.

Das Setzen eines einzelnen Schwellwerts entspricht dabei dem Aufruf von F_VN_Threshold [▶ 1363]. Dazu muss *fThresholdStep* auf 0 gesetzt werden und es wird nur der Wert von *fMinThreshold* verwendet.

Alternativ dazu können auch mehrere Schwellwerte berücksichtigt werden. Erster Schwellwert ist *fMinThreshold*. Von dem Wert ausgehend wird jeweils *fThresholdStep* für den nächsten Schwellwert addiert, bis der Schwellwert größer *fMaxThreshold* ist. Sollen zum Beispiel die drei Schwellwerte 100, 150 und 200 berücksichtigt werden, lautet die Parametrierung:

```
stBlobParams.fMinThreshold     := 100;
stBlobParams.fMaxThreshold     := 200;
stBlobParams.fThresholdStep    := 50;
```

Gleiche Konturen bei verschiedenen Schwellwerten

Es kann vorkommen, dass dieselbe Kontur bei verschiedenen Schwellwerten gefunden wird. Die Identifizierung gleicher Konturen erfolgt über einen Mittelpunktsvergleich. Dabei gibt *fMinBlobDistance* den minimalen Mittelpunktsabstand unterschiedlicher Konturen an. Ist der Abstand nicht erreicht, erfolgt entsprechend *eBlobCombination* (ETcVnBlobCombination [▶ 176]) die Konturauswahl.

Zudem kann das mehrfache Finden einer Kontur bei verschiedenen Schwellwerten auch ein Filterkriterium sein. Wie oft eine Kontur gefunden werden muss um berücksichtigt zu werden, wird über *nMinRepeatability* angegeben.

Filter

Im zweiten Schritt werden die gefundenen Konturen entsprechend der definierten Parameter gefiltert. Die einzelnen Filterkriterien können dabei jeweils über eine Bool-Variable aktiviert werden.

Fläche in Pixeln

- Parameter: bFilterByArea, fMinArea, fMaxArea
- Die Fläche einer Kontur kann alternativ über F_VN_ContourArea [▶ 988] ermittelt werden.

Zirkularität / Kreisförmigkeit

- Parameter: bFilterByCircularity, fMinCicularity, fMaxCircularity
- Wertebereich von 0 bis 1, wobei ein perfekter Kreis hat eine Zirkularität von 1 hat.
- Die Zirkularität / Kreisförmigkeit einer Kontur kann alternativ über F_VN_ContourCircularity [▶ 990] ermittelt werden.

Konvexität

- Parameter: bFilterByConvexity, fMinConvexity, fMaxConvexity
- Wertebereich von 0 bis 1, wobei eine vollkommen konvexe Form eine Konvexität von 1 hat.

Exzentrizität

- Parameter: bFilterByEccentricity, fMinEccentricity, fMaxEccentricity
- Wertebereich von 0 bis 1, wobei ein perfekter Kreis eine Exzentrizität von 0 hat.
- Die Exzentrizität einer Kontur kann alternativ über F_VN_ContourEccentricity [▶ 992] ermittelt werden.

Trägheitsverhältnis

- Parameter: bFilterByInertiaRatio, fMinInertiaRatio, fMaxInertiaRatio
- Wertebereich von 0 bis 1, wobei z. B. Formen mit gleicher Höhe und Breite ein Trägheitsverhältnis von 1 haben.

Anwendung

```
VAR
    stBlobParams    :   TcVnParamsBlobDetection;
END_VAR

stBlobParams.bFilterByArea  :=  TRUE;
stBlobParams.fMinArea       :=  100;
stBlobParams.fMaxArea       :=  100_000;

hr := F_VN_DetectBlobs(
    ipSrcImage      :=  ipImageIn,
    ipBlobContours  :=  ipContours,
    stParams        :=  stBlobsParams,
    hrPrev          :=  hr
);
```

Beispiel

- Blob Detection mit Watchdog Überwachung [▶ 2867]

Verwandte Funktionen

- F_VN_FindContours(Exp) [▶ 1216] zur allgemeinen Kontur-Findung
- F_VN_FindContourHierarchyExp [▶ 1212] mit Rückgabe der Hierarchie
- F_VN_DetectBlobs [▶ 1207] mit integrierter Filterung der Konturen

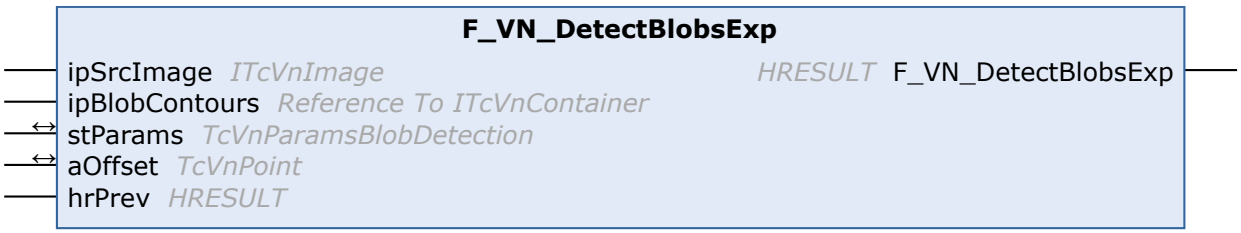
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.1.3 F_VN_DetectBlobsExp



Detects blob-contours. Applies a threshold, a contour search and offers several options for filtering the found contours. Provides easy setup for multiple thresholds and combination of results. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_DetectBlobsExp : HRESULT
VAR_INPUT
    ipSrcImage     : ITcVnImage;
    ipBlobContours : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    stParams       : TcVnParamsBlobDetection;
    aOffset        : TcVnPoint;
END_VAR
VAR_INPUT
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT or REAL, 1 channel or 3 channel (3 channel input is expected to be RGB and internally converted to Gray))
ipBlobContours	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the found contours (ContainerType_Vector_Vector_TcVnPoint2_DINT; The elements of this container are single contours. Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsBlobDetection [▶ 247]	Parameters to filter the detected contours.
aOffset	TcVnPoint [▶ 171]	Offset by which every contour point is shifted

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_DetectBlobsExp` ist die Expert-Variante von F_VN_DetectBlobs [▶ 1207]. Sie enthält zusätzliche Parameter.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1- oder 3-kanaliges Bild vom Typ `USINT` oder `REAL` sein. Dabei werden 3-kanalige Bilder als RGB-Bilder interpretiert und intern in ein Grauwertbild konvertiert.

Liste an gefundenen Konturen (Rückgabewert)

Alle im Eingangsbild gefundenen Konturen werden in dem Container `ipBlobContours` zurückgegeben.

Parameter zur Blob-Detektion

Die Parameter `stParams` der Struktur TcVnParamsBlobDetection [▶ 247] legen fest, welche Merkmale eine Kontur im Eingangsbild haben muss, um gefunden und zurückgegeben zu werden.

Offset (Expert)

Das Offset `aOffset` legt konstante X-/Y-Werte fest, um die alle Konturpunkte verschoben werden. Dies kann hilfreich sein, wenn Konturen in einer ROI gesucht werden und ihre Positionen sich auf das Originalbild beziehen sollen.

Verwandte Funktionen

- F_VN_FindContours(Exp) [▶ 1216] zur allgemeinen Kontur-Findung
- F_VN_FindContourHierarchyExp [▶ 1212] mit Rückgabe der Hierarchie
- F_VN_DetectBlobs [▶ 1207] mit integrierter Filterung der Konturen

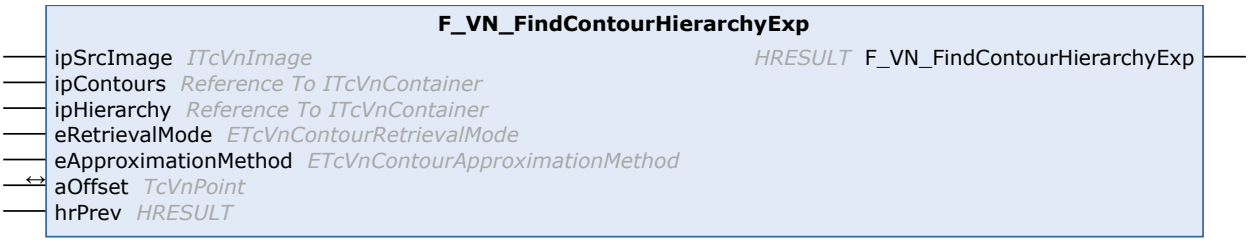
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.1.4 F_VN_FindContourHierarchyExp



Search for object contours in a binary image and determine their hierarchical relationship. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_FindContourHierarchyExp : HRESULT
VAR_INPUT
    ipSrcImage           : ITcVnImage;
    ipContours           : Reference To ITcVnContainer;
    ipHierarchy          : Reference To ITcVnContainer;
    eRetrievalMode       : ETcVnContourRetrievalMode;
    eApproximationMethod : ETcVnContourApproximationMethod;
END_VAR
VAR_IN_OUT
    aOffset              : TcVnPoint;
END_VAR
VAR_INPUT
    hrPrev               : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel, binary)
ipContours	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the found contours (ContainerType_Vector_Vector_TcVnPoint2_DINT; The elements of this container are single contours. Non-zero interface pointers are reused.)
ipHierarchy	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with information on the found contours' hierarchy (ContainerType_Vector_TcVnVector4_DINT; The elements of this container are four-dimensional vectors containing the 0-based indices of the next [0] and the previous contour [1] at the same level, the first child [2], and the parent [3]. Non-zero interface pointers are reused.)
eRetrievalMode	ETcVnContourRetrievalMode [▶ 202]	Specifies which contours are retrieved and how their relationship is encoded
eApproximationMethod	ETcVnContourApproximationMethod [▶ 199]	Contour encoding
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aOffset	TcVnPoint [▶ 171]	Offset by which every contour point is shifted

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_FindContourHierarchyExp` ist eine Erweiterung der Funktion F_VN_FindContoursExp [▶ 1219] um die Rückgabe der Konturhierarchie.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss genau 1 Kanal haben und sollte ein Binärbild sein.

Falls das Eingangsbild kein Binärbild ist, wird es intern mittels eines Schwellwerts von 1 in eins umgewandelt. Daher sollte das Eingangsbild manuell mit einem Schwellwert oder einer binären Segmentierung vorverarbeitet werden.

Liste an gefundenen Konturen (Rückgabewert)

Alle im Eingangsbild gefundenen Konturen werden in dem Container `ipContours` zurückgegeben. Der Container ist ein zweidimensionales Array von Punkten und hat daher den Typ `ContainerType_Vector_Vector_TcVnPoint2_DINT`.

Zurückgegeben werden die gefundenen Konturen in `ipContours`. Dabei setzt sich jede Kontur aus einer Summe von Punkten zusammen, welche in einem Container [▶ 163] zusammengefasst werden. Ein weiterer übergeordneter Container fasst dann nochmal alle Konturen zusammen.

Hierarchie (Expert, Rückgabewert)

Der Parameter `ipHierarchy` gibt die Hierarchie der gefundenen Konturen in einem Container [▶ 163] vom Typ TcVnVector4_DINT zurück. Die Vektoren des Containers haben folgende Elemente:

`[0]` – NEXT – Index der nächsten Kontur auf gleicher Ebene (gleichem Level)
`[1]` – PREVIOUS – Index der vorherigen Kontur auf gleicher Ebene (gleichem Level)
`[2]` – CHILD – Index der ersten Kind-Kontur
`[3]` – PARENT – Index der Eltern-Kontur

Suchmodus (Expert)

Der Suchmodus `eRetrievalMode` vom Typ ETcVnContourRetrievalMode [▶ 202] legt fest, nach welchen Konturen gesucht wird:

- `TCVN_CRM_LIST`: Alle Konturen
- `TCVN_CRM_EXTERNAL`: Nur Außenkonturen (wie bei F_VN_FindContours [▶ 1216])
- `TCVN_CRM_CONNECTED_COMPONENTS`: Konturen mit einer zweischichtigen Hierarchie
- `TCVN_CRM_TREE`: Alle Konturen; wird verwendet im Zusammenspiel mit der Funktion F_VN_FindContourHierarchyExp [▶ 1212], um die Hierarchie der Konturen zu bestimmen
- `TCVN_CRM_FLOODFILL`: Gibt die gefundenen Konturen mit Hilfe eines Floodfill-Algorithmus zurück (nur für DINT-Bilder verfügbar)

Approximationsart (Expert)

Die Approximationsart `eApproximationMethod` vom Typ ETcVnContourApproximationMethod [▶ 199] legt fest, ob die Konturen anhand aller umschließender Punkte beschrieben werden sollen (TCVN_CAM_NONE) oder ob die Anzahl der beschreibenden Punkte mit Hilfe von Approximation reduziert werden soll.

Offset (Expert)

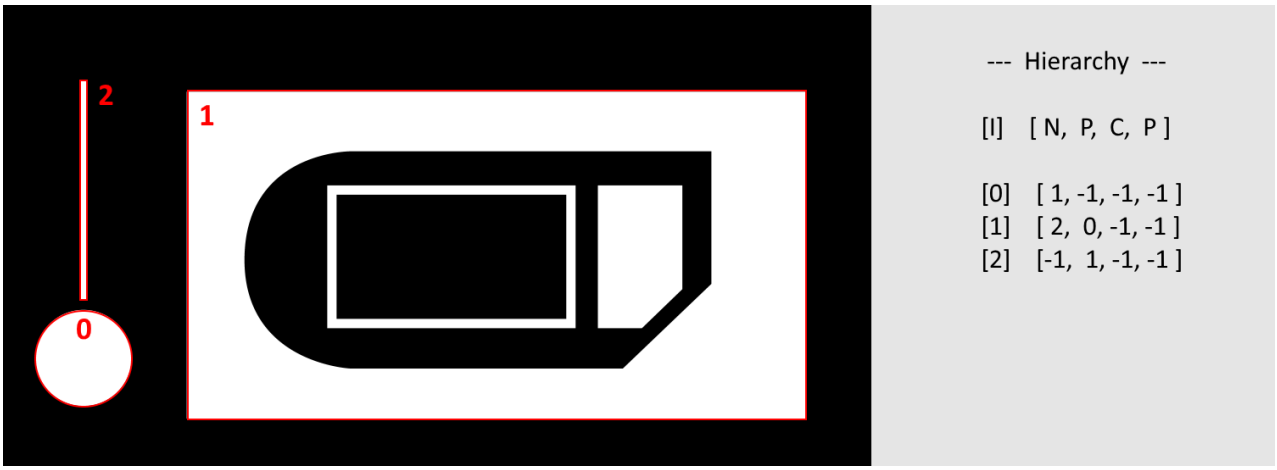
Das Offset `aOffset` legt konstante X-/Y-Werte fest, um die alle Konturpunkte verschoben werden. Dies kann hilfreich sein, wenn Konturen in einer ROI gesucht werden und ihre Positionen sich auf das Originalbild beziehen sollen.

Beispiele von Konturhierarchien

Über eRetrievalMode kann nicht nur unterschieden werden ob alle Konturen oder nur die äußeren gefunden werden sollen, sondern auch wie Konturen sortiert werden sollen und wie sie hierarchisch in Beziehung stehen. Dies kann am Parameter `ipHierarchy` abgelesen werden:

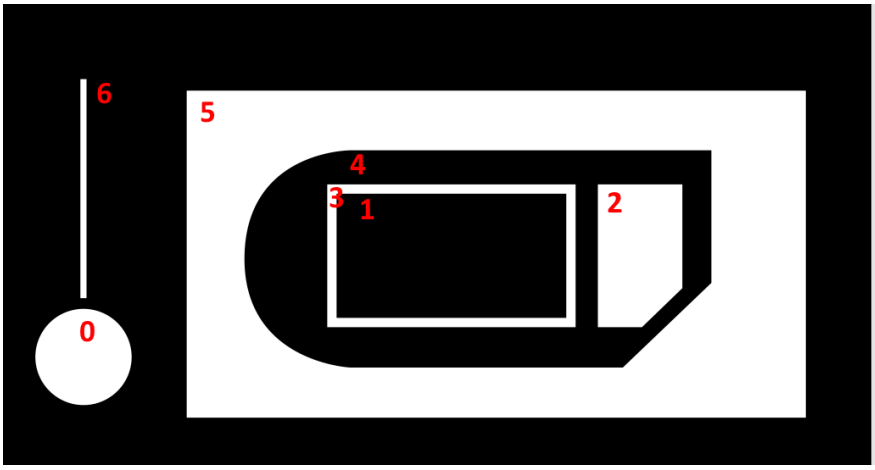
- **TCVN_CRM_EXTERNAL**

  Nur die äußeren Konturen, werden als Liste sortiert zurückgegeben. Die **C**hild `[2]` und **P**arent `[3]` Beziehungen werden nicht ermittelt und sind mit `-1` ungültig.



- **TCVN_CRM_LIST**

  Alle Konturen werden als Liste sortiert zurückgegeben. Die **C**hild `[2]` und **P**arent `[3]` Beziehungen werden nicht ermittelt und sind mit `-1` ungültig.



--- Hierarchy ---

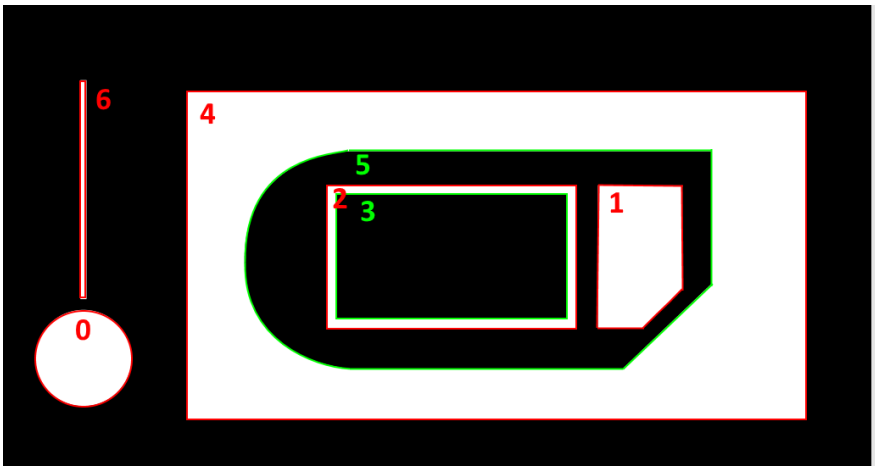
[I] [N, P, C, P]

[0] [1, -1, -1, -1]
[1] [2, 0, -1, -1]
[2] [3, 1, -1, -1]
[3] [4, 2, -1, -1]
[4] [5, 3, -1, -1]
[5] [6, 4, -1, -1]
[6] [-1, 4, -1, -1]

- **TCVN_CRM_CONNECTED_COMPONENTS**

  Alle Konturen werden nach einer 2-stufigen Hierarchie sortiert. Dabei werden zunächst die äußeren Kontouren betrachtet und dann in der zweiten Stufe die direkt darin liegende Kontur. Befinden sich weitere Kontouren innerhalb der zweiten Stufe werden diese, wie äußere Konturen auf die erste Stufe gesetzt

  In diesem Beispiel bedeutet das, dass die inneren **C**hild `[2]` Konturen (grün im Bild) nicht gleichzeitig **P**arent `[3]` Konturen (rot im Bild) sein können. Daher werden Konturen, welche bei einer Baumstruktur auf Level 3 liegen würden, wieder als äußere Konturen angesehen und auf Level 1 gesetzt (**P**arent `[3] = -1)`, hier Kontur 1 und 2.



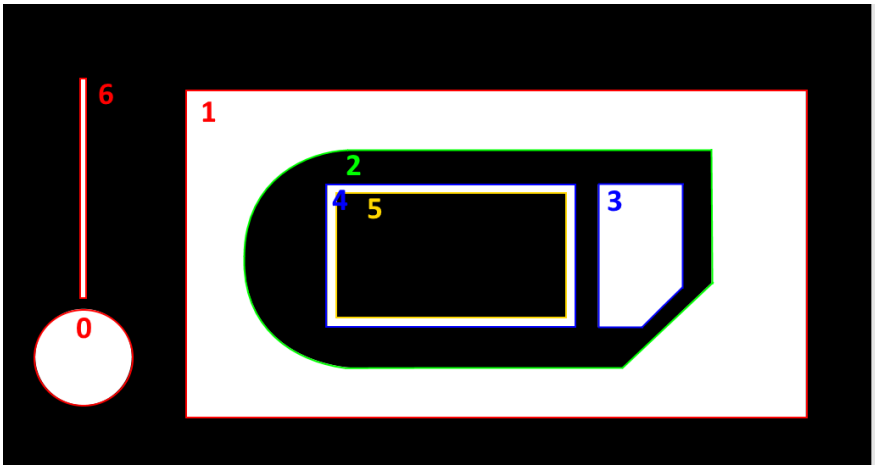
--- Hierarchy ---

[I] [N, P, C, P]

[0] [1, -1, -1, -1]
[1] [2, 0, -1, -1]
[2] [4, 1, 3, -1]
[3] [-1, -1, -1, 2]
[4] [6, 2, 5, -1]
[5] [-1, -1, -1, 4]
[6] [-1, 4, -1, -1]

- **TCVN_CRM_TREE**

  Alle Konturen werden nach einer Baumstruktur sortiert. Dabei ist es möglich, dass Konturen sowohl **CC**hild `[2]` als auch **P**arent `[3]` Konturen sind, je nachdem wie sie ineinander verschachtelt sind.



--- Hierarchy ---

[I] [N, P, C, P]

[0] [1, -1, -1, -1]
[1] [6, 0, 2, -1]
[2] [-1, -1, 3, 1]
[3] [4, -1, -1, 2]
[4] [-1, 3, 5, 2]
[5] [-1, -1, -1, 4]
[6] [-1, 1, -1, -1]

Beispiele

- Find Contour - Hierarchy & Retrieval Mode [▶ 2878], zum Parameter ETcVnContourRetrievalMode [▶ 202]
- Find Contour - Approximation Method [▶ 2875], zum Parameter ETcVnContourApproximationMethod [▶ 199]
- Find Contour anstelle der Blob Detection [▶ 2873], zum Vergleich zur Funktion F_VN_DetectBlobs [▶ 1207]

Verwandte Funktionen

- F_VN_FindContours(Exp) [▶ 1216] zur allgemeinen Kontur-Findung
- F_VN_FindContourHierarchyExp [▶ 1212] mit Rückgabe der Hierarchie
- F_VN_DetectBlobs [▶ 1207] mit integrierter Filterung der Konturen

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.1.5 F_VN_FindContours

Search for object contours in a binary image. Returns only external contours.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_FindContours : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
```

```
    ipContours : Reference To ITcVnContainer;
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel, binary)
ipContours	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the found contours (ContainerType_Vector_Vector_TcVnPoint2_DINT; The elements of this container are single contours. Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_FindContours` detektiert äußere Konturen im Eingangsbild.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss genau 1 Kanal haben und sollte ein Binärbild sein.

Falls das Eingangsbild kein Binärbild ist, wird es intern mittels eines Schwellwerts von 1 in eins umgewandelt. Daher sollte das Eingangsbild manuell mit einem Schwellwert oder einer binären Segmentierung vorverarbeitet werden.

Liste an gefundenen Konturen (Rückgabewert)

Alle im Eingangsbild gefundenen Konturen werden in dem Container `ipContours` zurückgegeben. Der Container ist ein zweidimensionales Array von Punkten und hat daher den Typ `ContainerType_Vector_Vector_TcVnPoint2_DINT`.

Zurückgegeben werden die gefundenen Konturen in `ipContours`. Dabei setzt sich jede Kontur aus einer Summe von Punkten zusammen, welche in einem Container [▶ 163] zusammengefasst werden. Ein weiterer übergeordneter Container fasst dann nochmal alle Konturen zusammen.

Expert-Parameter

Die Expert-Variante F_VN_FindContoursExp [▶ 1219] enthält zusätzliche Parameter. Damit können z. B. auch innere Konturen gefunden werden.

Anwendung

Das Finden und Verarbeiten von Konturen sieht z. B. so aus:

Vorverarbeitung

Da für das Eingangsbild `ipSrcImage` ein Binärbild verwendet werden soll, sollte vor dem Funktionsaufruf ein Schwellwert oder eine Binär-Segmentierung auf das Bild angewandt werden, z. B. mit der Funktion F_VN_Threshold [▶ 1363]:

```
hr := F_VN_Threshold(ipImageIn, ipImageWork, 128, 255, TCVN_TT_BINARY, hr);
```

Funktionsaufruf

Bei dem Funktionsaufruf ist nichts Weiteres zu beachten. Eine Einstellung von Parametern ist nur bei der Expert-Variante F_VN_FindContoursExp [▶ 1219] nötig.

```
hr := F_VN_FindContours(
    ipSrcImage  :=  ipImageIn,
    ipContours  :=  ipContours,
    hrPrev      :=  hr
);
```

Weiterverarbeitung

Mit F_VN_GetForwardIterator [▶ 788] kann ein Iterator auf den übergeordneten Container geholt werden, der es ermöglicht, nacheinander die Container der einzelnen Konturen über F_VN_GetContainer [▶ 786] abzurufen, um diese dann zum Beispiel mit den Contour Analysis Funktionen zu analysieren. Denken Sie dabei daran, den Iterators mit F_VN_IncrementIterator [▶ 791] zu inkrementieren.

```
hr := F_VN_GetForwardIterator(ipContours, ipIterator, hr);
WHILE SUCCEEDED(hr) AND_THEN ipIterator.CheckIfEnd() <> S_OK DO
    hr := F_VN_GetContainer( ipIterator, ipContour, hr);
    hr := F_VN_IncrementIterator(ipIterator, hr);

    // Analyse the single contour in ipContours with the contour analysis functions

END_WHILE
```

Alternativ können Sie eine einzelne Kontur auch über F_VN_GetAt_ITcVnContainer [▶ 665] aus dem Container aller Konturen abrufen.

Die Anzahl der gefundenen Konturen können Se über F_VN_GetNumberOfElements [▶ 789] abfragen.

Visualisierung

Zum Zeichnen der gefundenen Konturen verwenden Sie die Funktion F_VN_DrawContours(Exp) [▶ 1040].

```
hr := F_VN_DrawContours(
    ipContours      :=  ipContours,
    nContourIndex   :=  -1,
    ipDestImage     :=  ipImageRes,
    aColor          :=  aColorRed,
    nThickness      :=  5,
    hrPrev          :=  hr
);
```

Durch das Setzen von `nContoursIndex` auf `-1` werden alle Konturen in der Kontur-Liste `ipContours` gezeichnet.

Beispiele

- Find Contour anstelle der Blob Detection [▶ 2873]

Verwandte Funktionen

- F_VN_FindContours(Exp) [▶ 1216] zur allgemeinen Kontur-Findung
- F_VN_FindContourHierarchyExp [▶ 1212] mit Rückgabe der Hierarchie
- F_VN_DetectBlobs [▶ 1207] mit integrierter Filterung der Konturen

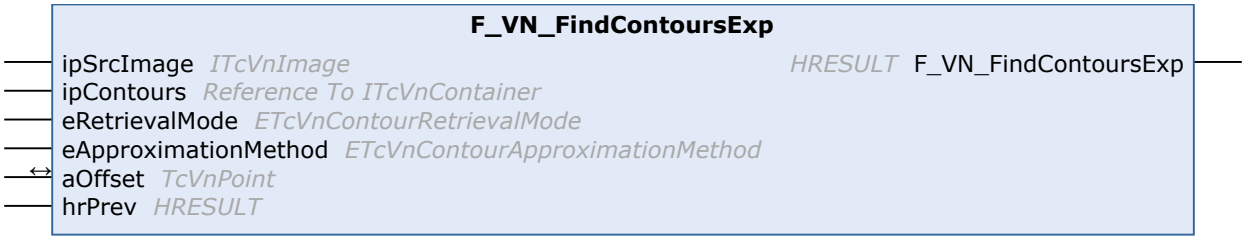
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.1.6 F_VN_FindContoursExp



Search for object contours in a binary image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_FindContoursExp : HRESULT
VAR_INPUT
    ipSrcImage           : ITcVnImage;
    ipContours           : Reference To ITcVnContainer;
    eRetrievalMode       : ETcVnContourRetrievalMode;
    eApproximationMethod : ETcVnContourApproximationMethod;
END_VAR
VAR_IN_OUT
    aOffset              : TcVnPoint;
END_VAR
VAR_INPUT
    hrPrev               : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel, binary)
ipContours	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the found contours (ContainerType_Vector_Vector_TcVnPoint2_DINT; The elements of this container are single contours. Non-zero interface pointers are reused.)
eRetrievalMode	ETcVnContourRetrievalMode [▶ 202]	Specifies which contours are retrieved and how their relationship is encoded
eApproximationMethod	ETcVnContourApproximationMethod [▶ 199]	Contour encoding
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aOffset	TcVnPoint [▶ 171]	Offset by which every contour point is shifted

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_FindContoursExp` ist die Expert-Variante von F_VN_FindContours [▶ 1216]. Sie enthält zusätzliche Parameter.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss genau 1 Kanal haben und sollte ein Binärbild sein.

Falls das Eingangsbild kein Binärbild ist, wird es intern mittels eines Schwellwerts von 1 in eins umgewandelt. Daher sollte das Eingangsbild manuell mit einem Schwellwert oder einer binären Segmentierung vorverarbeitet werden.

Liste an gefundenen Konturen (Rückgabewert)

Alle im Eingangsbild gefundenen Konturen werden in dem Container `ipContours` zurückgegeben. Der Container ist ein zweidimensionales Array von Punkten und hat daher den Typ `ContainerType_Vector_Vector_TcVnPoint2_DINT`.

Zurückgegeben werden die gefundenen Konturen in `ipContours`. Dabei setzt sich jede Kontur aus einer Summe von Punkten zusammen, welche in einem Container [▶ 163] zusammengefasst werden. Ein weiterer übergeordneter Container fasst dann nochmal alle Konturen zusammen.

Suchmodus (Expert)

Der Suchmodus `eRetrievalMode` vom Typ ETcVnContourRetrievalMode [▶ 202] legt fest, nach welchen Konturen gesucht wird:

- `TCVN_CRM_LIST`: Alle Konturen
- `TCVN_CRM_EXTERNAL`: Nur Außenkonturen (wie bei F_VN_FindContours [▶ 1216])
- `TCVN_CRM_CONNECTED_COMPONENTS`: Konturen mit einer zweischichtigen Hierarchie
- `TCVN_CRM_TREE`: Alle Konturen; wird verwendet im Zusammenspiel mit der Funktion F_VN_FindContourHierarchyExp [▶ 1212], um die Hierarchie der Konturen zu bestimmen
- `TCVN_CRM_FLOODFILL`: Gibt die gefundenen Konturen mit Hilfe eines Floodfill-Algorithmus zurück (nur für DINT-Bilder verfügbar)

Approximationsart (Expert)

Die Approximationsart `eApproximationMethod` vom Typ ETcVnContourApproximationMethod [▶ 199] legt fest, ob die Konturen anhand aller umschließender Punkte beschrieben werden sollen (TCVN_CAM_NONE) oder ob die Anzahl der beschreibenden Punkte mit Hilfe von Approximation reduziert werden soll.

Offset (Expert)

Das Offset `aOffset` legt konstante X-/Y-Werte fest, um die alle Konturpunkte verschoben werden. Dies kann hilfreich sein, wenn Konturen in einer ROI gesucht werden und ihre Positionen sich auf das Originalbild beziehen sollen.

Anwendung

Das Finden von allen Konturen im Eingangsbild, wobei die Konturpunkte nicht approximiert, aber in x-Richtung um 100px und in y-Richtung um -50px verschoben werden, sieht z. B. so aus:

```
VAR
    aOffset :   TcVnPoint   :=  [100, -50];
END_VAR

hr := F_VN_FindContours(
    ipSrcImage              :=  ipImageIn,
    ipContours              :=  ipContours,
    eRetrievalMode          :=  TCVN_CRM_LIST,
```

```
    eApproximationMethod    :=  TCVN_CAM_NONE,
    aOffset                 :=  aOffset,
    hrPrev                  :=  hr
);
```

Beispiele

- Find Contour - Hierarchy & Retrieval Mode [▶ 2878], zum Parameter ETcVnContourRetrievalMode [▶ 202]
- Find Contour - Approximation Method [▶ 2875], zum Parameter ETcVnContourApproximationMethod [▶ 199]
- Find Contour anstelle der Blob Detection [▶ 2873], zum Vergleich zur Funktion F_VN_DetectBlobs [▶ 1207]

Verwandte Funktionen

- F_VN_FindContours(Exp) [▶ 1216] zur allgemeinen Kontur-Findung
- F_VN_FindContourHierarchyExp [▶ 1212] mit Rückgabe der Hierarchie
- F_VN_DetectBlobs [▶ 1207] mit integrierter Filterung der Konturen

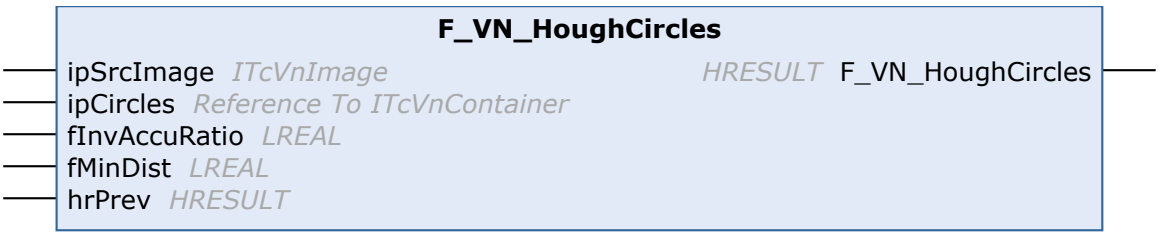
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.1.7 F_VN_HoughCircles



Search for circles using the Hough transform.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_HoughCircles : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipCircles     : Reference To ITcVnContainer;
    fInvAccuRatio : LREAL;
    fMinDist      : LREAL;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, 1 channel, gray-level)
ipCircles	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the found circles (ContainerType_Vector_TcVnVector3_REAL; Each container element contains the x coordinate of the circle center [0], the y coordinate of the circle center [1], and the radius [2], respectively. Non-zero interface pointers are reused.)
fInvAccuRatio	LREAL	Inverted ratio of the accumulator size in relation to the source image's size (must be > 0. A value of 2 means that the size is halved in both directions.)
fMinDist	LREAL	Smallest allowed distance of two circles (must be > 0)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

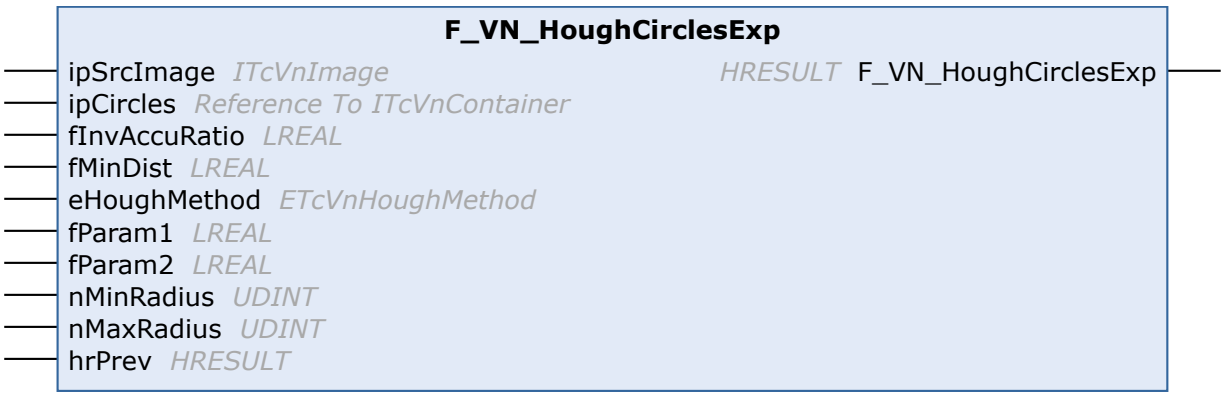
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.1.8 F_VN_HoughCirclesExp



Search for circles using the Hough transform. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_HoughCirclesExp : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipCircles     : Reference To ITcVnContainer;
    fInvAccuRatio : LREAL;
    fMinDist      : LREAL;
    eHoughMethod  : ETcVnHoughMethod;
    fParam1       : LREAL;
```

```
    fParam2       : LREAL;
    nMinRadius    : UDINT;
    nMaxRadius    : UDINT;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, 1 channel, gray-level)
ipCircles	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the found circles (ContainerType_Vector_TcVnVector3_REAL; Each container element contains the x coordinate of the circle center [0], the y coordinate of the circle center [1], and the radius [2], respectively. Non-zero interface pointers are reused.)
fInvAccuRatio	LREAL	Inverted ratio of the accumulator size in relation to the source image's size (must be > 0. A value of 2 means that the size is halved in both directions.)
fMinDist	LREAL	Smallest allowed distance of two circles (must be > 0)
eHoughMethod	ETcVnHoughMethod [▶ 216]	Hough method to use (GRADIENT or GRADIENT_ALT)
fParam1	LREAL	First method specific parameter (GRADIENT, GRADIENT_ALT: upper threshold for canny edge detection, which must be > 0)
fParam2	LREAL	Second method specific parameter (GRADIENT: accumulator threshold for detecting circle centers, which must be > 0. GRADIENT_ALT: required circle perfectness, which must be > 0 and < 1, 1 would be a perfect circle)
nMinRadius	UDINT	Minimum circle radius allowed
nMaxRadius	UDINT	Maximum circle radius allowed (if 0, the value is internally set to max(image rows, image columns))
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

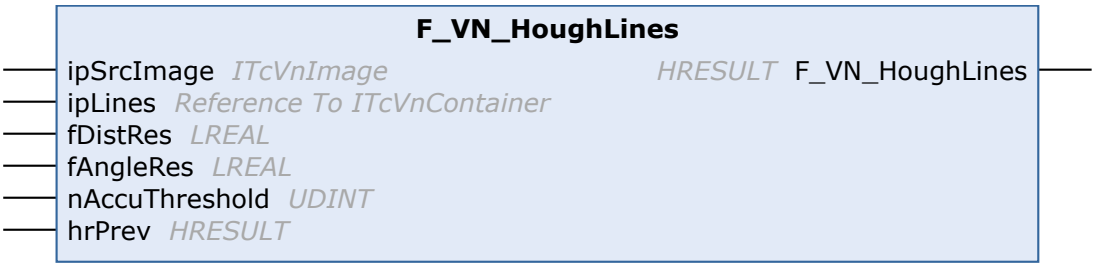
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.1.9 F_VN_HoughLines



Search for lines using the standard Hough transform.

Syntax

Definition:

```
FUNCTION F_VN_HoughLines : HRESULT
VAR_INPUT
    ipSrcImage     : ITcVnImage;
    ipLines        : Reference To ITcVnContainer;
    fDistRes       : LREAL;
    fAngleRes      : LREAL;
    nAccuThreshold : UDINT;
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, 1 channel, binary, may be modified by the function)
ipLines	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the found lines (ContainerType_Vector_TcVnVector2_REAL; Each container element contains the distance from the origin [0] in pixels and the rotation angle [1] in radians, respectively. Non-zero interface pointers are reused.)
fDistRes	LREAL	Distance resolution of the accumulator (in pixels, must be > 0)
fAngleRes	LREAL	Angle resolution of the accumulator (in radians, must be > 0)
nAccuThreshold	UDINT	Accumulator threshold
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

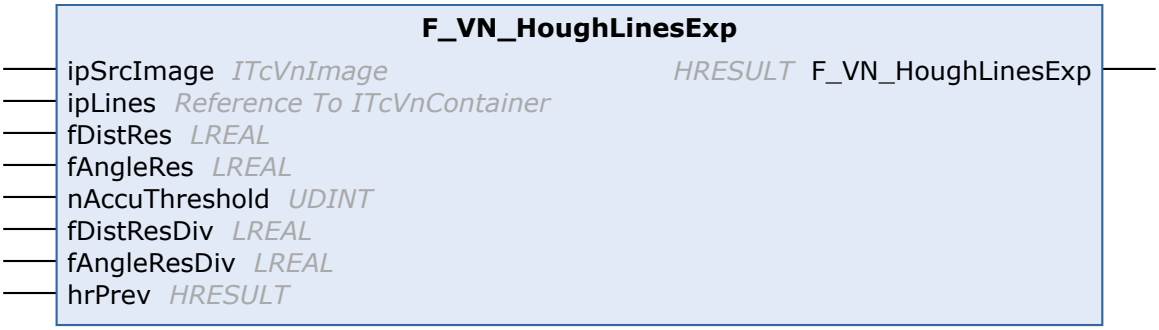
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.1.10 F_VN_HoughLinesExp



Search for lines using the standard Hough transform. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_HoughLinesExp : HRESULT
VAR_INPUT
    ipSrcImage     : ITcVnImage;
    ipLines        : Reference To ITcVnContainer;
    fDistRes       : LREAL;
    fAngleRes      : LREAL;
    nAccuThreshold : UDINT;
    fDistResDiv    : LREAL;
    fAngleResDiv   : LREAL;
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, 1 channel, binary, may be modified by the function)
ipLines	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the found lines (ContainerType_Vector_TcVnVector2_REAL; Each container element contains the distance from the origin [0] in pixels and the rotation angle [1] in radians, respectively. Non-zero interface pointers are reused.)
fDistRes	LREAL	Distance resolution of the accumulator (in pixels, must be > 0)
fAngleRes	LREAL	Angle resolution of the accumulator (in radians, must be > 0)
nAccuThreshold	UDINT	Accumulator threshold
fDistResDiv	LREAL	Divisor of the distance resolution for the multi-scale Hough transform (>= 0, must be > 0 if fAngleResDiv is > 0)
fAngleResDiv	LREAL	Divisor of the angle resolution for the multi-scale Hough transform (>= 0, must be > 0 if fDistResDiv is > 0)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

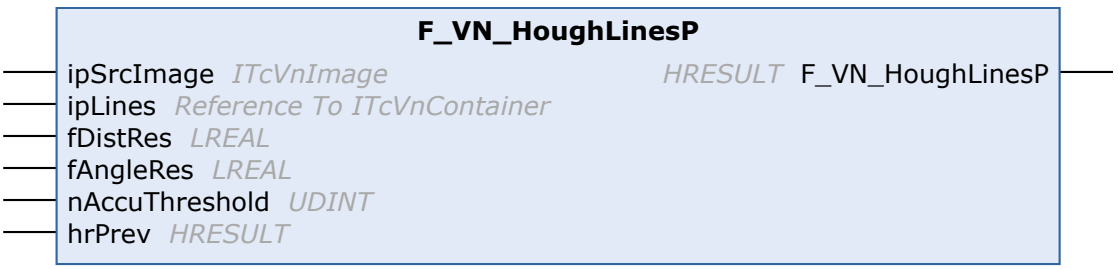
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.1.11 F_VN_HoughLinesP



Search for line segments using the probabilistic Hough transform.

Syntax

Definition:

```
FUNCTION F_VN_HoughLinesP : HRESULT
VAR_INPUT
    ipSrcImage     : ITcVnImage;
    ipLines        : Reference To ITcVnContainer;
    fDistRes       : LREAL;
    fAngleRes      : LREAL;
    nAccuThreshold : UDINT;
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, 1 channel, binary, may be modified by the function)
ipLines	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the found line segments (ContainerType_Vector_TcVnVector4_DINT; Each container element contains the two ending points. Non-zero interface pointers are reused.)
fDistRes	LREAL	Distance resolution of the accumulator (in pixels, must be > 0)
fAngleRes	LREAL	Angle resolution of the accumulator (in radians, must be > 0)
nAccuThreshold	UDINT	Accumulator threshold
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

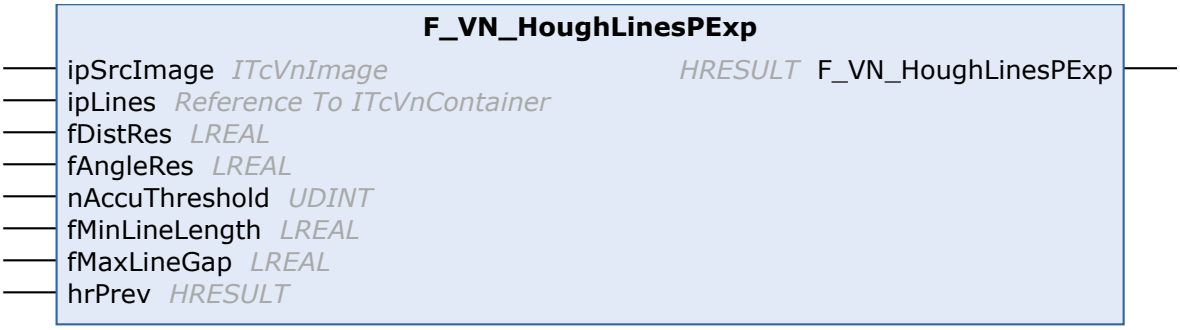
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.1.12 F_VN_HoughLinesPExp



Search for line segments using the probabilistic Hough transform. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_HoughLinesPExp : HRESULT
VAR_INPUT
    ipSrcImage     : ITcVnImage;
    ipLines        : Reference To ITcVnContainer;
    fDistRes       : LREAL;
    fAngleRes      : LREAL;
    nAccuThreshold : UDINT;
    fMinLineLength : LREAL;
    fMaxLineGap    : LREAL;
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, 1 channel, binary, may be modified by the function)
ipLines	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the found line segments (ContainerType_Vector_TcVnVector4_DINT; Each container element contains the two ending points. Non-zero interface pointers are reused.)
fDistRes	LREAL	Distance resolution of the accumulator (in pixels, must be > 0)
fAngleRes	LREAL	Angle resolution of the accumulator (in radians, must be > 0)
nAccuThreshold	UDINT	Accumulator threshold
fMinLineLength	LREAL	Minimum line length to search for
fMaxLineGap	LREAL	Maximum gap between points on the same line
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

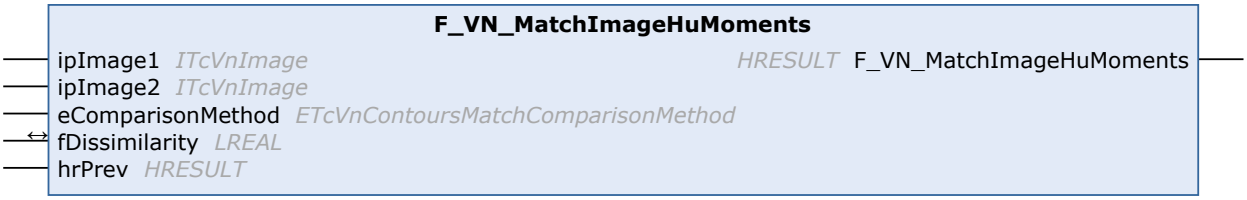
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.1.13 F_VN_MatchImageHuMoments



Compare two images using the Hu moment invariants.

Syntax

Definition:

```
FUNCTION F_VN_MatchImageHuMoments : HRESULT
VAR_INPUT
    ipImage1          : ITcVnImage;
    ipImage2          : ITcVnImage;
    eComparisonMethod : ETcVnContoursMatchComparisonMethod;
END_VAR
VAR_IN_OUT
    fDissimilarity    : LREAL;
END_VAR
VAR_INPUT
    hrPrev            : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage1	ITcVnImage [▶ 426]	First image (1 channel)
ipImage2	ITcVnImage [▶ 426]	Second image (1 channel)
eComparisonMethod	ETcVnContoursMatchComparisonMethod [▶ 204]	Method used for comparing the Hu moment invariants of the images
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fDissimilarity	LREAL	Returns the dissimilarity of the image Hu moment invariants depending on the chosen comparison method

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.1.14 F_VN_MatchTemplate

Match a template image with every location in the source image (using the TCVN_TMM_CCORR_NORMED method) and save the comparison results.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_MatchTemplate : HRESULT
VAR_INPUT
    ipSrcImage      : ITcVnImage;
    ipTemplateImage : ITcVnImage;
    ipResultImage   : Reference To ITcVnImage;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT or REAL, 1 or 3 channels)
ipTemplateImage	ITcVnImage [▶ 426]	Template image (same type as ipSrcImage, smaller width and height)
ipResultImage	Reference To ITcVnImage [▶ 426]	Returns the result image (REAL, normalized to [0..1], 1 channel, dimensions: (ipSrcImage.width - ipTemplateImage.width + 1) x (ipSrcImage.height - ipTemplateImage.height + 1). The best match is the global maximum. The position in ipResultImage is the top-left corner of ipTemplateImage position in ipSrcImage)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

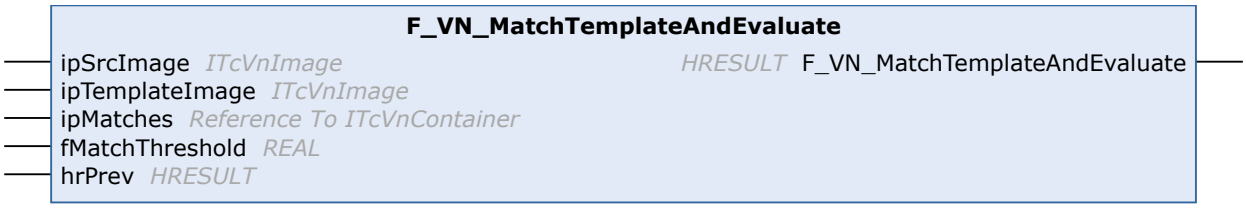
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.1.15 F_VN_MatchTemplateAndEvaluate



Match a template image with every location in the source image (using the TCVN_TMM_CCORR_NORMED method) and evaluate the comparison results. Returns a sorted list of possible matches (best match first). Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_MatchTemplateAndEvaluate : HRESULT
VAR_INPUT
    ipSrcImage      : ITcVnImage;
    ipTemplateImage : ITcVnImage;
    ipMatches       : Reference To ITcVnContainer;
```

```
    fMatchThreshold : REAL;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT or REAL, 1 or 3 channels)
ipTemplateImage	ITcVnImage [▶ 426]	Template image (same type as ipSrcImage, smaller width and height)
ipMatches	Reference To ITcVnContainer [▶ 385]	Returns the matching positions (ContainerType_Vector_TcVnPoint2_DINT, where each element represents the top-left corner of ipTemplateImage) in ipSrcImage, sorted by relevance (best match first)
fMatchThreshold	REAL	Threshold to separate relevant from irrelevant matches (0..1, 1.0 would be a perfect match. To find a suitable value, you could evaluate some sample result images of F_VN_MatchTemplate.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

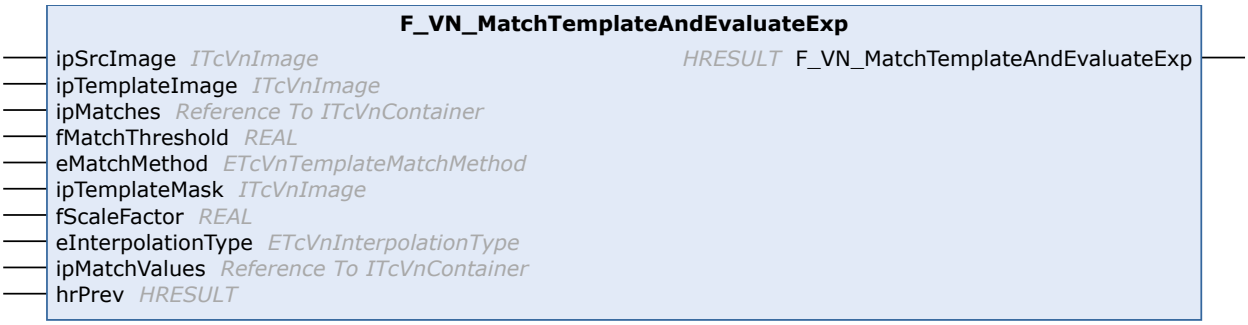
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.1.16 F_VN_MatchTemplateAndEvaluateExp



Match a template image with every location in the source image and evaluate the comparison results.
Returns a sorted list of possible matches (best match first). (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_MatchTemplateAndEvaluateExp : HRESULT
VAR_INPUT
    ipSrcImage         : ITcVnImage;
    ipTemplateImage    : ITcVnImage;
    ipMatches          : Reference To ITcVnContainer;
```

```
    fMatchThreshold    : REAL;
    eMatchMethod       : ETcVnTemplateMatchMethod;
    ipTemplateMask     : ITcVnImage;
    fScaleFactor       : REAL;
    eInterpolationType : ETcVnInterpolationType;
    ipMatchValues      : Reference To ITcVnContainer;
    hrPrev             : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT or REAL, 1 or 3 channels)
ipTemplateImage	ITcVnImage [▶ 426]	Template image (same type as ipSrcImage, smaller width and height)
ipMatches	Reference To ITcVnContainer [▶ 385]	Returns the matching positions (ContainerType_Vector_TcVnPoint2_DINT, where each element represents the top-left corner of ipTemplateImage) in ipSrcImage, sorted by relevance (best match first)
fMatchThreshold	REAL	Threshold to separate relevant from irrelevant matches (0..1 for NORMED methods, otherwise dependent on template size and content. To find a suitable value, you could evaluate some sample result images of F_VN_MatchTemplateExp.)
eMatchMethod	ETcVnTemplateMatchMethod [▶ 231]	Specifies the template match method
ipTemplateMask	ITcVnImage [▶ 426]	Optional mask for ipTemplateImage (same type and size as ipTemplateImage)
fScaleFactor	REAL	Factor (0..1] to reduce source and template image width and height for better performance (but less accuracy!)
eInterpolationType	ETcVnInterpolationType [▶ 217]	Image resize interpolation type (only used if fScaleFactor != 1, TCVN_IT_BILINEAR recommended for most cases)
ipMatchValues	Reference To ITcVnContainer [▶ 385]	Optionally returns the matching values (ContainerType_Vector_REAL, same size and sort order as ipMatches. Set to 0 if not required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.1.17 F_VN_MatchTemplateExp

Match a template image with every location in the source image and save the comparison results. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_MatchTemplateExp : HRESULT
VAR_INPUT
    ipSrcImage      : ITcVnImage;
    ipTemplateImage : ITcVnImage;
    ipResultImage   : Reference To ITcVnImage;
    eMatchMethod    : ETcVnTemplateMatchMethod;
    ipTemplateMask  : ITcVnImage;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT or REAL, 1 or 3 channels)
ipTemplateImage	ITcVnImage [▶ 426]	Template image (same type as ipSrcImage, smaller width and height)
ipResultImage	Reference To ITcVnImage [▶ 426]	Returns the result image (REAL, 1 channel, dimensions: (ipSrcImage.width - ipTemplateImage.width + 1) x (ipSrcImage.height - ipTemplateImage.height + 1). The best match is the global minimum (SQDIFF(_NORMED)) or maximum (CCORR(_NORMED), CCOEFF(_NORMED)). The position in ipResultImage is the top-left corner of ipTemplateImage position in ipSrcImage)
eMatchMethod	ETcVnTemplateMatchMethod [▶ 231]	Specifies the template match method
ipTemplateMask	ITcVnImage [▶ 426]	Optional mask for ipTemplateImage (same type and size as ipTemplateImage)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

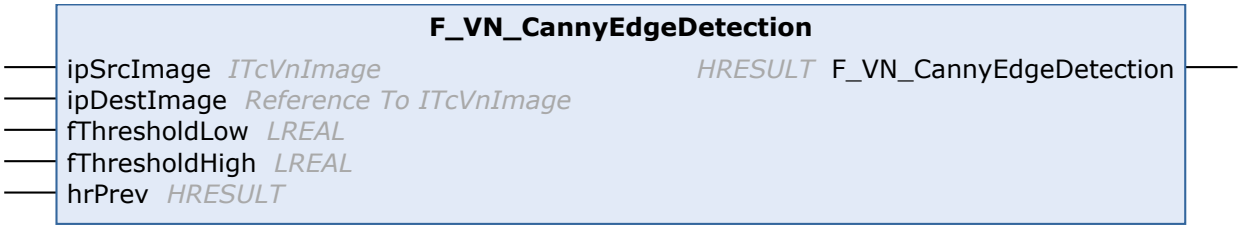
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.2 F_VN_CannyEdgeDetection



Find edges using the Canny edge detection algorithm.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_CannyEdgeDetection : HRESULT
VAR_INPUT
    ipSrcImage     : ITcVnImage;
    ipDestImage    : Reference To ITcVnImage;
    fThresholdLow  : LREAL;
    fThresholdHigh : LREAL;
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (elements of type USINT)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (elements of type USINT. An appropriate destination image will be created if required.)
fThresholdLow	LREAL	Low threshold
fThresholdHigh	LREAL	High threshold
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Diese Funktion findet Kanten auf einem Bild und stellt diese durch weiße Linien auf schwarzem Hintergrund dar. Zudem existiert diese Expert-Variante F_VN_CannyEdgeDetectionExp [▶ 1235] mit erweiterten Parametriermöglichkeiten.

Algorithmus

Die Canny Edge Kantendetektion ist ein mehrstufiger Algorithmus, um möglichst robust Kanten zu detektieren. Er besteht aus folgenden Schritten:

1. Reduzierung von Rauschen mittels Gauß-Filter.
2. Kanten finden mittels Sobel-Operator.
3. Linienverdünnung mittels Non-maximum suppression, d. h. die resultierenden Linien sind 1px dick.

4. Hysterese, so dass nur starke Linien erhalten bleiben (siehe Schwellwert-Parameter unten).

Parameter

Originalbild

Für das Originalbild `ipSrcImage` wird ein 1-kanaliges 8-Bit Intensitätsbild (Pixelformat `TCVN_ET_USINT`) erwartet. Sie können also ein Grauwertbild oder einzelne Kanäle eines Farbbilds verwenden.

Ergebnisbild

Das Ergebnisbild `ipDestImage` besitzt dasselbe Format.

Schwellwerte

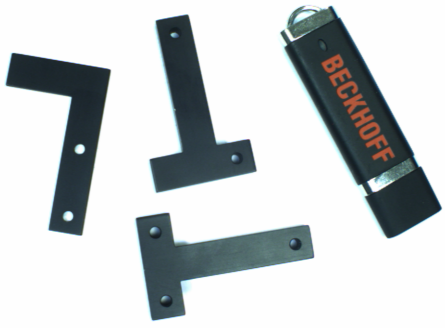
Die beiden Schwellwerte `fThresholdLow` und `fThresholdHigh` bestimmen, welche der gefundenen Kanten akzeptiert werden: Pixel mit einem Gradienten höher als `fThresholdHigh` werden akzeptiert. Pixel mit einem Farbverlauf unterhalb von `fThresholdLow` werden verworfen. Pixel mit Gradienten dazwischen werden nur akzeptiert, wenn sie mit bereits akzeptierten Pixeln verbunden sind.

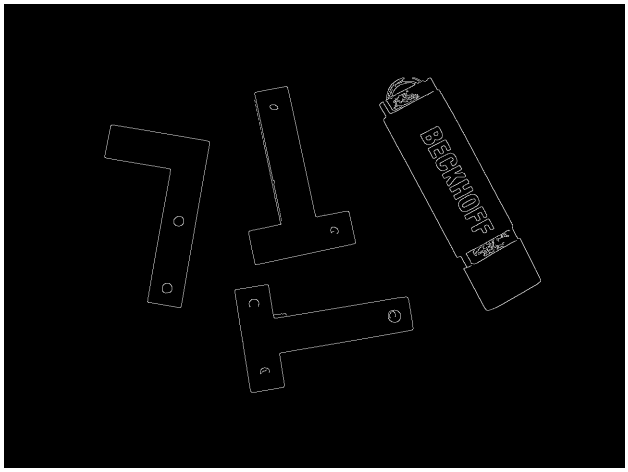
Durch die beiden Schwellwerte ist eine flexible Einstellung möglich. Eine harte Schwelle ist ebenfalls realisierbar, indem Sie beide Werte gleichsetzen.

Wenn Sie die Schwellwerte zu hoch einstellen, können wichtige Kanteninformationen verloren gehen. Bei zu niedrigen Schwellwerten können Rauschen und andere irrelevante Informationen als Kanten interpretiert werden.

Anwendung

```
hr := F_VN_CannyEdgeDetection(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageWork,
    fThresholdLow   :=  0,
    fThresholdHigh  :=  200,
    hr
);
```





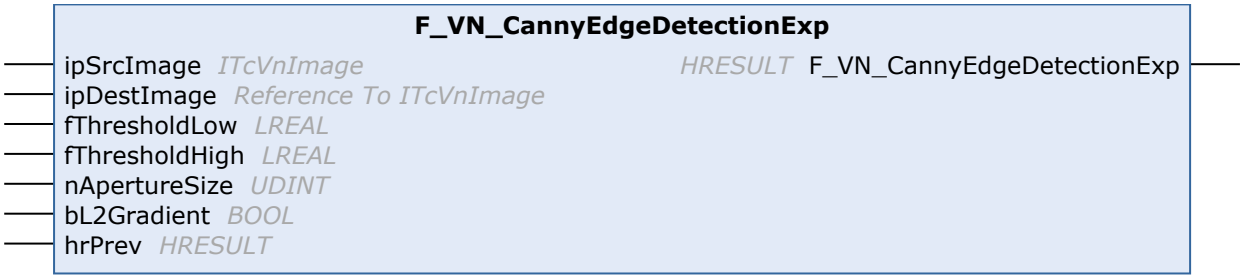
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.3 F_VN_CannyEdgeDetectionExp



Find edges using the Canny edge detection algorithm. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_CannyEdgeDetectionExp : HRESULT
VAR_INPUT
    ipSrcImage     : ITcVnImage;
    ipDestImage    : Reference To ITcVnImage;
    fThresholdLow  : LREAL;
    fThresholdHigh : LREAL;
    nApertureSize  : UDINT;
    bL2Gradient    : BOOL;
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (elements of type USINT)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (elements of type USINT. An appropriate destination image will be created if required.)
fThresholdLow	LREAL	Low threshold
fThresholdHigh	LREAL	High threshold
nApertureSize	UDINT	Aperture size for the Sobel operator (3, 5, 7)
bL2Gradient	BOOL	If true, the more accurate (and slower) L2 norm is used instead of the L1 norm
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Diese Funktion ist die Expert-Variante von F_VN_CannyEdgeDetection [▶ 1233]. Sie enthält erweiterte Parametriermöglichkeiten.

Parameter

Folgende Parameter stehen in dieser Expert-Variante zusätzlich zur Verfügung:

Blendengröße

Für `nApertureSize` sind nur ungerade Werte ab `3` valide.

L1-/L2-Norm

Für die Berechnung der Gradienten stehen zwei Normen zur Verfügung. Die L1-Norm ist schneller. Die L2-Norm ist hingegen genauer.

$$L_1 = \left|\frac{dI}{dx}\right| + \left|\frac{dI}{dy}\right| \qquad L_2 = \sqrt{\left(\frac{dI}{dx}\right)^2 + \left(\frac{dI}{dy}\right)^2}$$

Setzen Sie den Parameter `bL2Gradient` auf `FALSE`, um die L1-Norm zu verwenden, und auf `TRUE`, um die L2-Norm zu verwenden.

Anwendung

Eine Anwendung dieser Funktion kann z.B. wie folgt aussehen:

```
hr := F_VN_CannyEdgeDetectionExp(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageWork,
    fThresholdLow   :=  0,
    fThresholdHigh  :=  200,
    nApertureSize   :=  5,
    bL2Gradient     :=  TRUE,
    hr
);
```

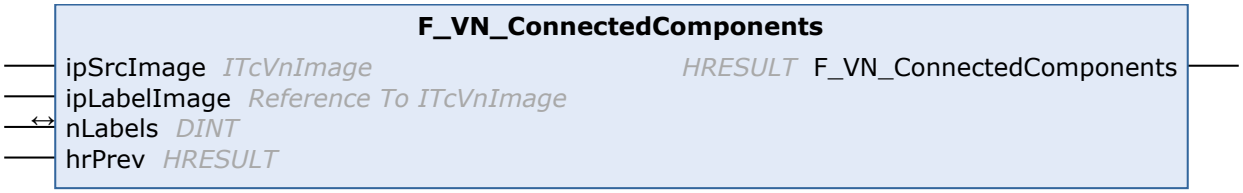
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.4 F_VN_ConnectedComponents



Computes the connected components of an image.

Syntax

Definition:

```
FUNCTION F_VN_ConnectedComponents : HRESULT
VAR_INPUT
    ipSrcImage   : ITcVnImage;
    ipLabelImage : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    nLabels      : DINT;
END_VAR
VAR_INPUT
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, 1 channel, binary)
ipLabelImage	Reference To ITcVnImage [▶ 426]	Returns the labels for each source image pixel (1 channel, DINT. An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nLabels	DINT	Returns the number of labels

Return value

HRESULT [▶ 150]

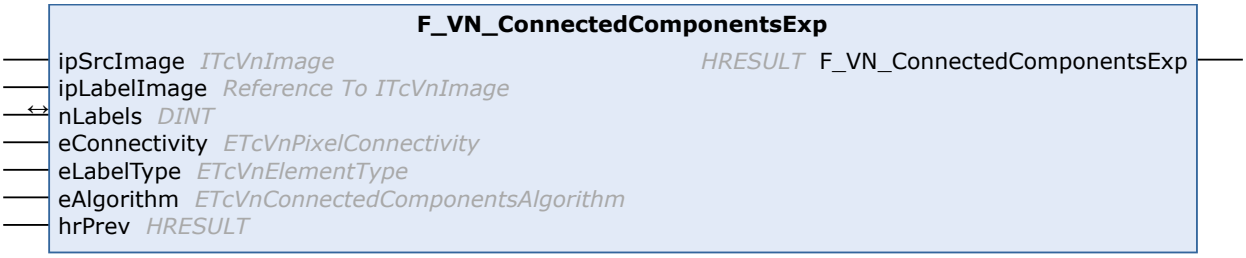
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.5 F_VN_ConnectedComponentsExp



Computes the connected components of an image. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_ConnectedComponentsExp : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipLabelImage  : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    nLabels       : DINT;
END_VAR
VAR_INPUT
    eConnectivity : ETcVnPixelConnectivity;
    eLabelType    : ETcVnElementType;
    eAlgorithm    : ETcVnConnectedComponentsAlgorithm;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, 1 channel, binary)
ipLabelImage	Reference To ITcVnImage [▶ 426]	Returns the labels for each source image pixel (1 channel, type depends on eLabelType. An appropriate destination image will be created if required.)
eConnectivity	ETcVnPixelConnectivity [▶ 224]	Selects if 4- or 8-way pixel connectivity should be used
eLabelType	ETcVnElementType [▶ 210]	Selects the type of ipLabelImage (only UINT or DINT supported)
eAlgorithm	ETcVnConnectedComponentsAlgorithm [▶ 198]	Selects the applied algorithm
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nLabels	DINT	Returns the number of labels

Return value

HRESULT [▶ 150]

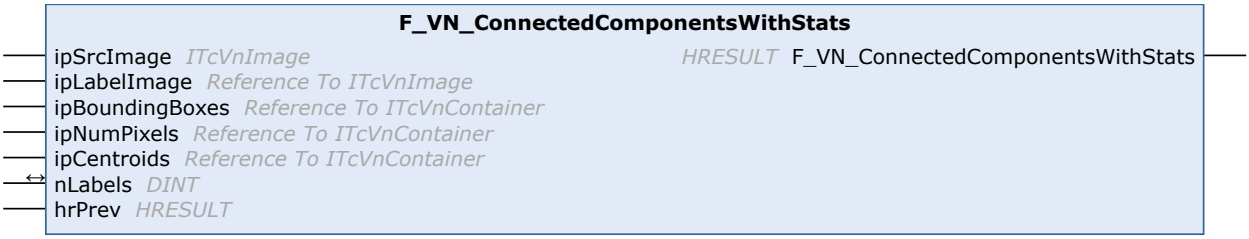
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.6 F_VN_ConnectedComponentsWithStats



Computes the connected components and corresponding statistics of an image.

Syntax

Definition:

```
FUNCTION F_VN_ConnectedComponentsWithStats : HRESULT
VAR_INPUT
    ipSrcImage      : ITcVnImage;
    ipLabelImage    : Reference To ITcVnImage;
    ipBoundingBoxes : Reference To ITcVnContainer;
    ipNumPixels     : Reference To ITcVnContainer;
    ipCentroids     : Reference To ITcVnContainer;
END_VAR
```

```
VAR_IN_OUT
    nLabels         : DINT;
END_VAR
VAR_INPUT
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, 1 channel, binary)
ipLabelImage	Reference To ITcVnImage [▶ 426]	Returns the labels for each source image pixel (1 channel, DINT. An appropriate destination image will be created if required.)
ipBoundingBoxes	Reference To ITcVnContainer [▶ 385]	Returns the bounding boxes for each labeled region. (ContainerType_Vector_TcVnRectangle_DINT; Non-zero interface pointers are reused.)
ipNumPixels	Reference To ITcVnContainer [▶ 385]	Returns the number of pixels for each labeled region. (ContainerType_Vector_DINT; Non-zero interface pointers are reused.)
ipCentroids	Reference To ITcVnContainer [▶ 385]	Returns the centroids for each labeled region. (ContainerType_Vector_TcVnPoint2_LREAL; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nLabels	DINT	Returns the number of labels

Return value

HRESULT [▶ 150]

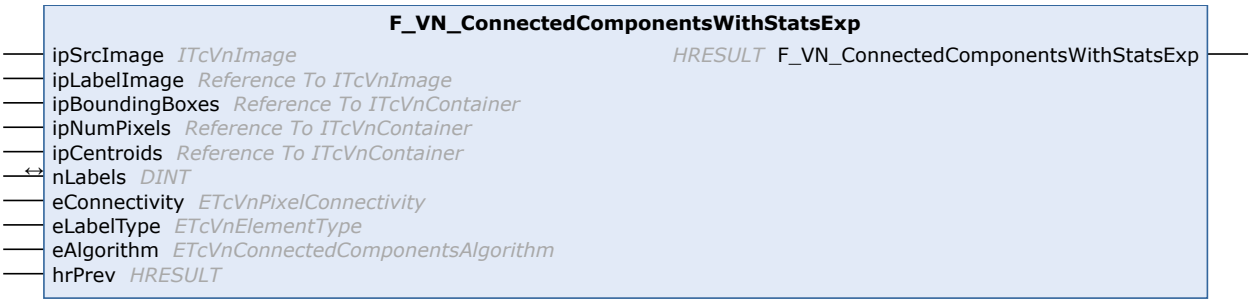
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.7 F_VN_ConnectedComponentsWithStatsExp



Computes the connected components and corresponding statistics of an image. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_ConnectedComponentsWithStatsExp : HRESULT
VAR_INPUT
    ipSrcImage      : ITcVnImage;
    ipLabelImage    : Reference To ITcVnImage;
    ipBoundingBoxes : Reference To ITcVnContainer;
    ipNumPixels     : Reference To ITcVnContainer;
    ipCentroids     : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    nLabels         : DINT;
END_VAR
VAR_INPUT
    eConnectivity   : ETcVnPixelConnectivity;
    eLabelType      : ETcVnElementType;
    eAlgorithm      : ETcVnConnectedComponentsAlgorithm;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, 1 channel, binary)
ipLabelImage	Reference To ITcVnImage [▶ 426]	Returns the labels for each source image pixel (1 channel, DINT. An appropriate destination image will be created if required.)
ipBoundingBoxes	Reference To ITcVnContainer [▶ 385]	Returns the bounding boxes for each labeled region. (ContainerType_Vector_TcVnRectangle_DINT; Non-zero interface pointers are reused.)
ipNumPixels	Reference To ITcVnContainer [▶ 385]	Returns the number of pixels for each labeled region. (ContainerType_Vector_DINT; Non-zero interface pointers are reused.)
ipCentroids	Reference To ITcVnContainer [▶ 385]	Returns the centroids for each labeled region. (ContainerType_Vector_TcVnPoint2_LREAL; Non-zero interface pointers are reused.)
eConnectivity	ETcVnPixelConnectivity [▶ 224]	Selects if 4- or 8-way pixel connectivity should be used
eLabelType	ETcVnElementType [▶ 210]	Selects the type of ipLabelImage (only UINT or DINT supported)
eAlgorithm	ETcVnConnectedComponentsAlgorithm [▶ 198]	Selects the applied algorithm
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nLabels	DINT	Returns the number of labels

Return value

HRESULT [▶ 150]

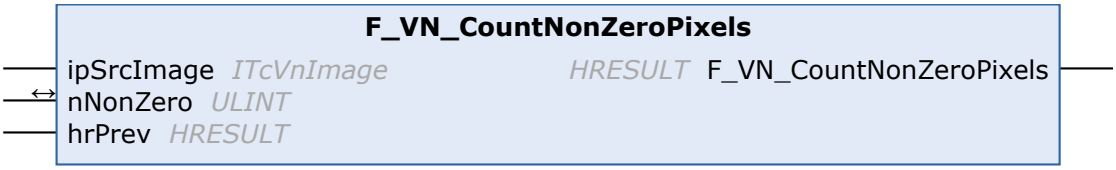
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.8 F_VN_CountNonZeroPixels



Counts the non-zero pixels in a single-channel image (e.g. useful to analyze threshold results).

Syntax

Definition:

```
FUNCTION F_VN_CountNonZeroPixels : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    nNonZero   : ULINT;
END_VAR
VAR_INPUT
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nNonZero	ULINT	Returns the number of non-zero pixels in ipSrcImage

Return value

HRESULT [▶ 150]

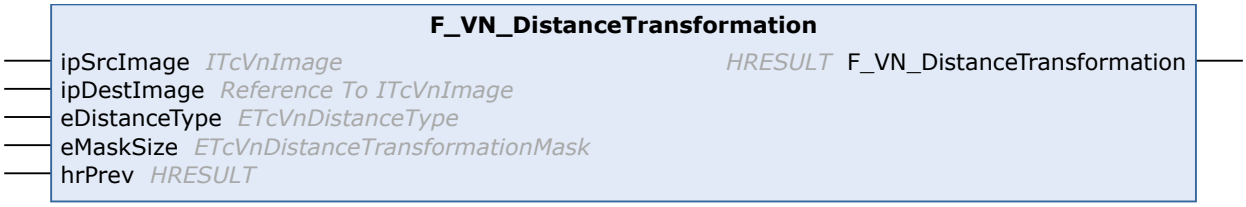
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.9 F_VN_DistanceTransformation



Calculates the distance transformation, which is the distance to the closest zero pixel in a binary image.

Syntax

Definition:

```
FUNCTION F_VN_DistanceTransformation : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipDestImage   : Reference To ITcVnImage;
    eDistanceType : ETcVnDistanceType;
    eMaskSize     : ETcVnDistanceTransformationMask;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (REAL, 1 channel. An appropriate destination image will be created if required.)
eDistanceType	ETcVnDistanceType [▶ 207]	Distance computation method (supported: L1, L2, C)
eMaskSize	ETcVnDistanceTransformationMask [▶ 206]	Size of the distance transformation mask
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Diese Funktion führt die Distanz-Transformation eines Bildes durch. Zudem existiert die Expert-Variante F_VN_DistanceTransformationExp [▶ 1244] mit erweiterten Parametriermöglichkeiten.

Parameter

Folgende Parameter stehen in der Basis-Variante zur Verfügung:

Originalbild

Für das Originalbild `ipSrcImage` wird ein 1-kanaliges 8-Bit Bild (Element-Typ `TCVN_ET_USINT`) erwartet. Andere Bildformate müssen zunächst umgewandelt werden.

Ergebnisbild

Das Ergebnisbild `ipDestImage` wird mit Element-Typ `TCVN_ET_REAL` zurückgegeben. Der Wert jedes Pixels beschreibt dabei die Distanz des jeweiligen Pixels zum nächstgelegenen null-wertigen Pixel im Originalbild.

Anzeige im ADS Image Watch

Um das Ergebnisbild im ADS Image Watch anzeigen zu können, muss es zunächst in ein kompatibles Bildformat umgewandelt werden. Je nach Anwendungsfall kann zudem eine Skalierung der Pixelwerte sinnvoll sein.

Distanz-Typ

Der Distanz-Typ `eDistanceType` bestimmt, mit welcher Formel die Distanz zwischen einem Pixel und dem nächstgelegenen null-wertigen Pixel berechnet wird. Aktuell werden folgende Distanztypen aus dem Enum ETcVnDistanceType [▶ 207] unterstützt:

- TCVN_DT_L1
- TCVN_DT_L2
- TCVN_DT_C

Masken-Größe

Die Maskengröße `eMaskSize` beschreibt die Annäherung der euklidischen Distanz, wenn als Distanztyp die L2-Norm ausgewählt ist:

- TCVN_DTM_PRECISE: Der euklidische Abstand wird präsize berechnet.
- TCVN_DTM_3: Der euklidische Abstand wird als Summe von Abstandselementen innerhalb einer 3x3-Maske angenähert (horizontal, vertikal und diagonal).
- TCVN_DTM_5: Der euklidische Abstand wird als Summe von Abstandselementen innerhalb einer 5x5-Maske angenähert (horizontal, vertikal, diagonal und Springerzug (wie beim Schach)).

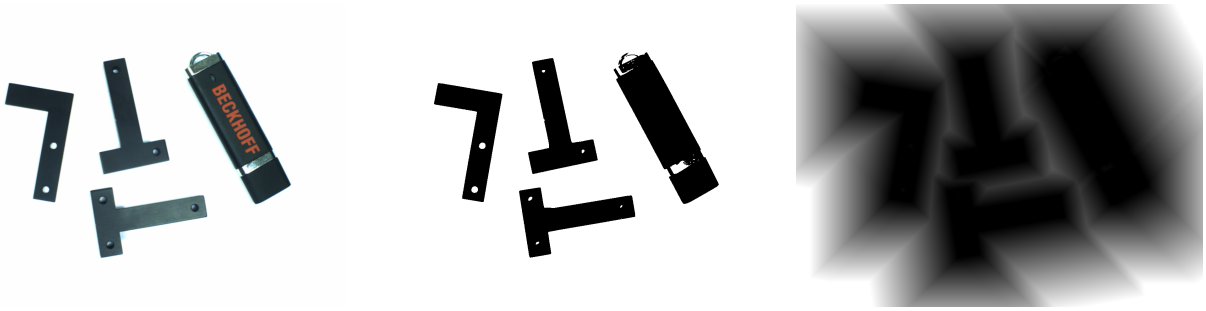
Bei den Distanztypen L1 und C hat die Maskengröße keinen Effekt. Stattdessen wird dann intern immer eine 3x3-Maske verwendet.

Anwendung

Eine Anwendung dieser Funktion kann z. B. wie folgt aussehen:

```
// convert to binary image
hr := F_VN_Threshold(ipImageIn, ipImageWork, 128, 255, TCVN_TT_BINARY, hr);
// apply distance transformation
hr := F_VN_DistanceTransformation(
    ipSrcImage      :=  ipImageWork,
    ipDestImage     :=  ipImageWork,
    eDistanceType   :=  TCVN_DT_C,
    eMaskSize       :=  TCVN_DTM_3,
    hr
);
// convert image to display it in ADS Image Watch
hr := F_VN_ConvertElementType(ipImageWork, ipImageWork, TCVN_ET_USINT, hr);
```

Originalbild **Binärbild** **Ergebnisbild**



Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.10 F_VN_DistanceTransformationExp

Calculates the distance transformation, which is the distance to the closest zero pixel in a binary image. Additionally, an image with component labels is created (discrete Voronoi diagram). (expert function)

Syntax

Definition:

```
FUNCTION F_VN_DistanceTransformationExp : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipDestImage   : Reference To ITcVnImage;
    eDistanceType : ETcVnDistanceType;
    eMaskSize     : ETcVnDistanceTransformationMask;
    ipDestLabels  : Reference To ITcVnImage;
    eLabelType    : ETcVnDistanceTransformationLabel;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (REAL, 1 channel. An appropriate destination image will be created if required.)
eDistanceType	ETcVnDistanceType [▶ 207]	Distance computation method (supported: L1, L2, C)
eMaskSize	ETcVnDistanceTransformationMask [▶ 206]	Size of the distance transformation mask (PRECISE not supported for label computation)
ipDestLabels	Reference To ITcVnImage [▶ 426]	Returns the component labels (Discrete Voronoi diagram; DINT, 1 channel. An appropriate image will be created if required.)
eLabelType	ETcVnDistanceTransformationLabel [▶ 206]	Type of the labels
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

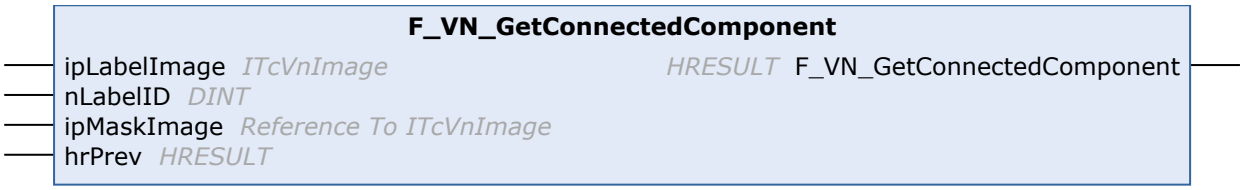
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.11 F_VN_GetConnectedComponent



Get the mask image of a connected component from a label image based on the label ID.

Syntax

Definition:

```
FUNCTION F_VN_GetConnectedComponent : HRESULT
VAR_INPUT
    ipLabelImage : ITcVnImage;
    nLabelID     : DINT;
    ipMaskImage  : Reference To ITcVnImage;
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipLabelImage	ITcVnImage [▶ 426]	Image containing labels for each pixel (1 channel DINT or UINT).
nLabelID	DINT	Value of the requested label
ipMaskImage	Reference To ITcVnImage [▶ 426]	Returns a mask image marking the pixels that belong to the component (1 channel USINT.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

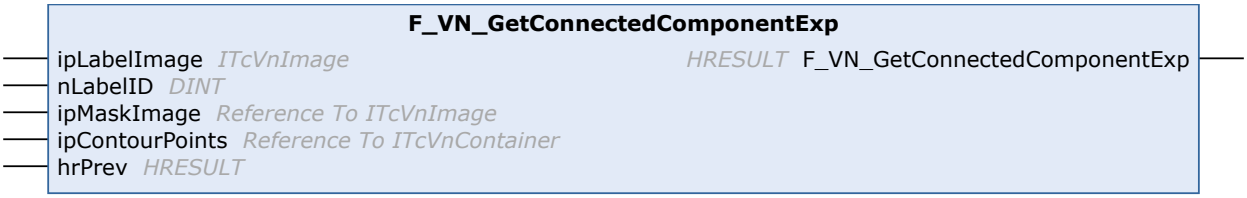
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.12 F_VN_GetConnectedComponentExp



Get the mask image (and optionally the outer contour) of a connected component from a label image based on the label ID. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_GetConnectedComponentExp : HRESULT
VAR_INPUT
    ipLabelImage    : ITcVnImage;
    nLabelID        : DINT;
    ipMaskImage     : Reference To ITcVnImage;
    ipContourPoints : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipLabelImage	ITcVnImage [▶ 426]	Image containing labels for each pixel (1 channel DINT or UINT).
nLabelID	DINT	Value of the requested label
ipMaskImage	Reference To ITcVnImage [▶ 426]	Returns a mask image marking the pixels that belong to the component (1 channel USINT. Mask is optional, set to 0 if not required.)
ipContourPoints	Reference To ITcVnContainer [▶ 385]	Returns the outer contour of the component(ContainerType_Vector_TcVnPoint2_DINT. Contour is optional, set to 0 if not required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

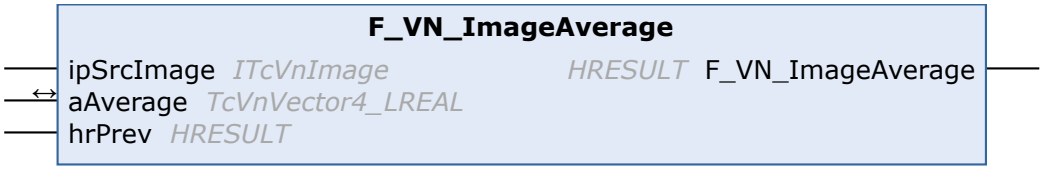
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.13 F_VN_ImageAverage



Computes the (channel-wise) average pixel value of an image.

Syntax

Definition:

```
FUNCTION F_VN_ImageAverage : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aAverage   : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 - 4 channels)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAverage	TcVnVector4_LREAL [▶ 173]	Returns the (channel-wise) average pixel value of ipSrcImage

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.14 F_VN_ImageAverageExp

Computes the (channel-wise) average pixel value of an image. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_ImageAverageExp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aAverage   : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipMask     : ITcVnImage;
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 - 4 channels)
ipMask	ITcVnImage [▶ 426]	Mask of type USINT (1 channel)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAverage	TcVnVector4_LREAL [▶ 173]	Returns the (channel-wise) average pixel value of ipSrcImage

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.15 F_VN_ImageAverageStdDev

Computes the (channel-wise) average pixel value and the corresponding standard deviation of an image.

Syntax

Definition:

```
FUNCTION F_VN_ImageAverageStdDev : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aAverage   : TcVnVector4_LREAL;
    aStdDev    : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 - 4 channels)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAverage	TcVnVector4_LREAL [▶ 173]	Returns the (channel-wise) average pixel value of ipSrcImage
aStdDev	TcVnVector4_LREAL [▶ 173]	Returns the (channel-wise) pixel value standard deviation of ipSrcImage

Return value

HRESULT [▶ 150]

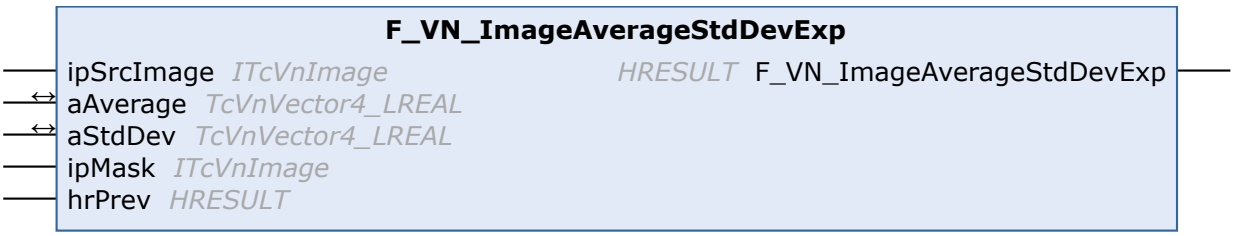
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.16 F_VN_ImageAverageStdDevExp



Computes the (channel-wise) average pixel value and the corresponding standard deviation of an image. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_ImageAverageStdDevExp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
```

```
END_VAR
VAR_IN_OUT
    aAverage    : TcVnVector4_LREAL;
    aStdDev     : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipMask      : ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 - 4 channels)
ipMask	ITcVnImage [▶ 426]	Mask of type USINT (1 channel)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAverage	TcVnVector4_LREAL [▶ 173]	Returns the (channel-wise) average pixel value of ipSrcImage
aStdDev	TcVnVector4_LREAL [▶ 173]	Returns the (channel-wise) pixel value standard deviation of ipSrcImage

Return value

HRESULT [▶ 150]

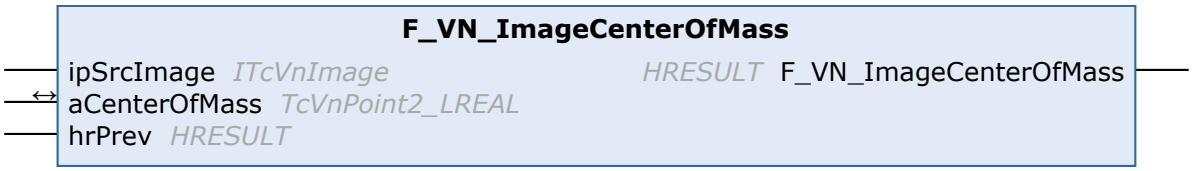
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.17 F_VN_ImageCenterOfMass



Computes the center of mass of an image.

Syntax

Definition:

```
FUNCTION F_VN_ImageCenterOfMass : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
END_VAR
VAR_IN_OUT
    aCenterOfMass : TcVnPoint2_LREAL;
END_VAR
```

```
VAR_INPUT
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCenterOfMass	TcVnPoint2_LREAL [▶ 171]	Returns the center of mass of the image

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Diese Funktion berechnet den Schwerpunkt des Eingangsbildes. Es existiert die Expert-Variante F_VN_ImageCenterOfMassExp [▶ 1252] mit erweiterten Parametriermöglichkeiten.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss 1-kanalig sein. Wenn Sie den Schwerpunkt eines mehrkanaligen Bildes berechnen wollen, haben Sie folgende Möglichkeiten:

- Sie können das mehrkanalige Bild mit F_VN_SplitImageChannels [▶ 834] in einzelne Kanäle aufteilen und für jeden Kanal einzeln einen Schwerpunkt berechnen.
- Sie können das mehrkanalige Bild mit F_VN_ConvertColorSpace [▶ 1270] in ein 1-kanaliges Bild umwandeln und dafür einen Schwerpunkt berechnen.

Schwerpunkt (Rückgabewert)

Über die Referenz `aCenterOfMass` wird der berechnete Schwerpunkt zurückgegeben.

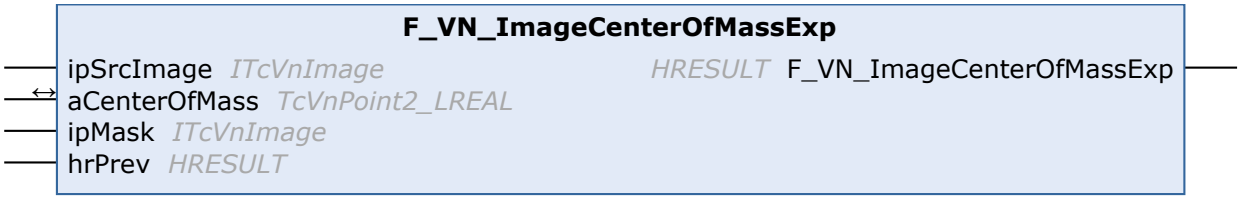
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.18 F_VN_ImageCenterOfMassExp



Computes the center of mass of an image.

Syntax

Definition:

```
FUNCTION F_VN_ImageCenterOfMassExp : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
END_VAR
VAR_IN_OUT
    aCenterOfMass : TcVnPoint2_LREAL;
END_VAR
VAR_INPUT
    ipMask        : ITcVnImage;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel)
ipMask	ITcVnImage [▶ 426]	Optional mask (1 channel of type USINT, same width and height as ipSrcImage)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aCenterOfMass	TcVnPoint2_LREAL [▶ 171]	Returns the center of mass of the image

Return value

HRESULT [▶ 150]

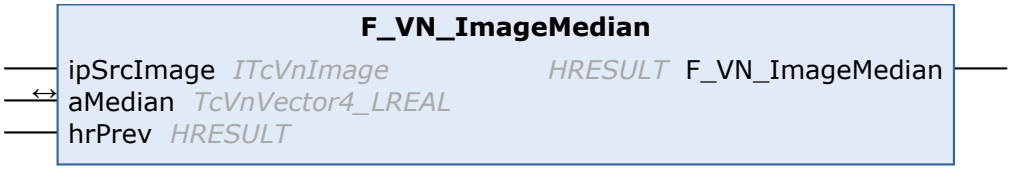
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.19 F_VN_ImageMedian



Computes the (approximated, channel-wise) median pixel value of an image.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_ImageMedian : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aMedian    : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 - 4 channels)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector4_LREAL [▶ 173]	Returns the (approximated, channel-wise) median pixel value of ipSrcImage

Return value

HRESULT [▶ 150]

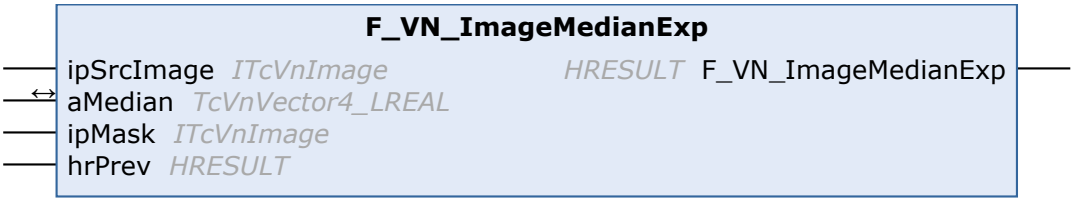
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.20 F_VN_ImageMedianExp



Computes the (approximated, channel-wise) median pixel value of an image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_ImageMedianExp : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    aMedian    : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipMask     : ITcVnImage;
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 - 4 channels)
ipMask	ITcVnImage [▶ 426]	Mask of type USINT (1 channel)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMedian	TcVnVector4_LREAL [▶ 173]	Returns the (approximated, channel-wise) median pixel value of ipSrcImage

Return value

HRESULT [▶ 150]

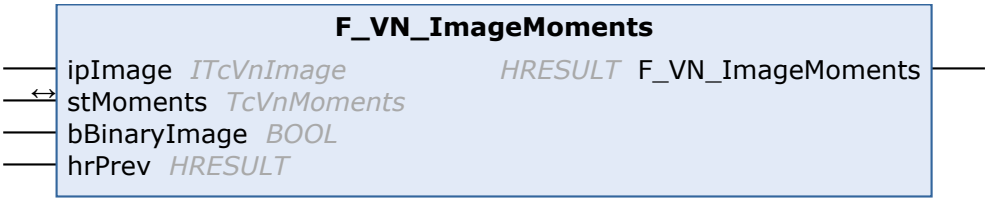
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.21 F_VN_ImageMoments



Computes the spatial moments, the central moments, and the central normalized moments of an image up to the third order.

Syntax

Definition:

```
FUNCTION F_VN_ImageMoments : HRESULT
VAR_INPUT
    ipImage      : ITcVnImage;
END_VAR
VAR_IN_OUT
    stMoments    : TcVnMoments;
END_VAR
VAR_INPUT
    bBinaryImage : BOOL;
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Source image (1 channel, elements of type USINT or REAL)
bBinaryImage	BOOL	If true, all non-zero pixels are treated as ones
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stMoments	TcVnMoments [▶ 245]	Returns a struct containing the moments

Return value

HRESULT [▶ 150]

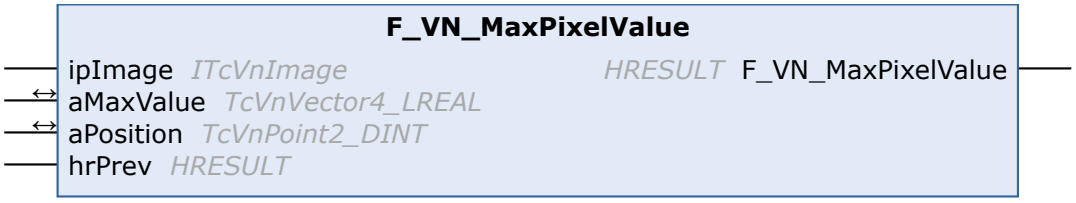
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.22 F_VN_MaxPixelValue



Finds the maximum pixel value in an image (1 - 4 channels supported).

Syntax

Definition:

```
FUNCTION F_VN_MaxPixelValue : HRESULT
VAR_INPUT
    ipImage   : ITcVnImage;
END_VAR
VAR_IN_OUT
    aMaxValue : TcVnVector4_LREAL;
    aPosition : TcVnPoint2_DINT;
END_VAR
VAR_INPUT
    hrPrev    : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Source image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMaxValue	TcVnVector4_LREAL [▶ 173]	Returns the maximum pixel value
aPosition	TcVnPoint2_DINT [▶ 171]	Returns the first found position of aMaxValue

Return value

HRESULT [▶ 150]

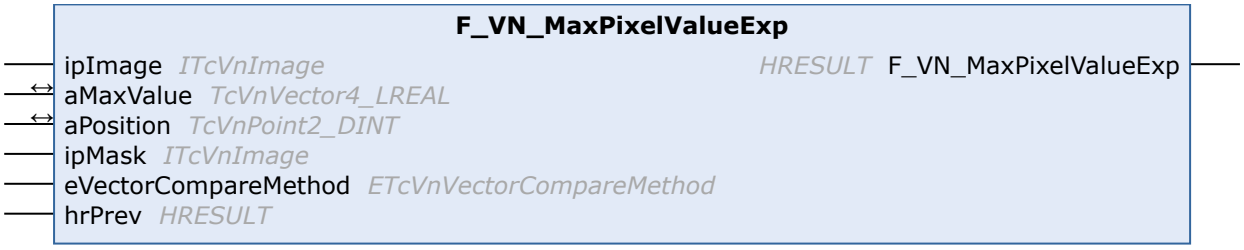
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.23 F_VN_MaxPixelValueExp



Finds the maximum pixel value in an image (1 - 4 channels supported). (expert function)

Syntax

Definition:

```
FUNCTION F_VN_MaxPixelValueExp : HRESULT
VAR_INPUT
    ipImage              : ITcVnImage;
END_VAR
VAR_IN_OUT
    aMaxValue            : TcVnVector4_LREAL;
    aPosition            : TcVnPoint2_DINT;
END_VAR
VAR_INPUT
    ipMask               : ITcVnImage;
    eVectorCompareMethod : ETcVnVectorCompareMethod;
    hrPrev               : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Source image
ipMask	ITcVnImage [▶ 426]	Optional mask to specify which pixel positions are considered (USINT, set parameter to 0 if not required)
eVectorCompareMethod	ETcVnVectorCompareMethod [▶ 234]	Select a vector compare method for multi-channel images
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMaxValue	TcVnVector4_LREAL [▶ 173]	Returns the maximum pixel value
aPosition	TcVnPoint2_DINT [▶ 171]	Returns the first found position of aMaxValue (not supported for multi-channel images with ELEMENTWISE)

Return value

HRESULT [▶ 150]

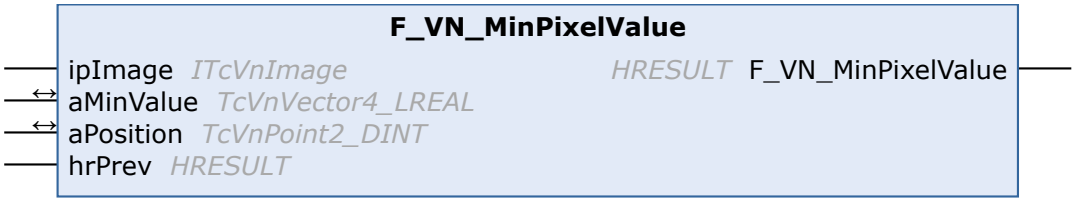
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.24 F_VN_MinPixelValue



Finds the minimum pixel value in an image (1 - 4 channels supported).

Syntax

Definition:

```
FUNCTION F_VN_MinPixelValue : HRESULT
VAR_INPUT
    ipImage    : ITcVnImage;
END_VAR
VAR_IN_OUT
    aMinValue : TcVnVector4_LREAL;
    aPosition : TcVnPoint2_DINT;
END_VAR
VAR_INPUT
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Source image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMinValue	TcVnVector4_LREAL [▶ 173]	Returns the minimum pixel value
aPosition	TcVnPoint2_DINT [▶ 171]	Returns the first found position of aMinValue

Return value

HRESULT [▶ 150]

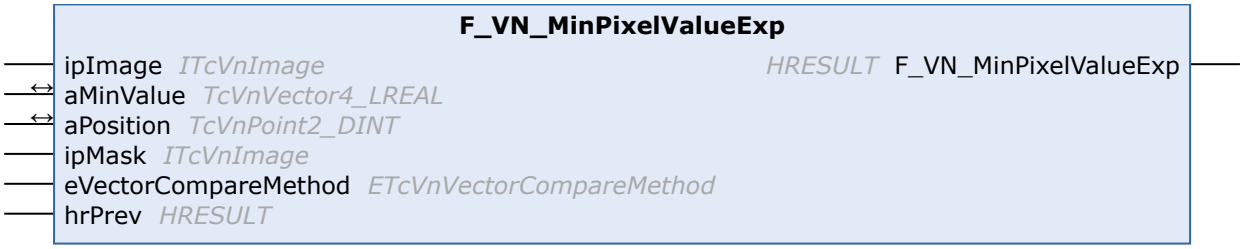
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.25 F_VN_MinPixelValueExp



Finds the minimum pixel value in an image (1 - 4 channels supported). (expert function)

Syntax

Definition:

```
FUNCTION F_VN_MinPixelValueExp : HRESULT
VAR_INPUT
    ipImage              : ITcVnImage;
END_VAR
VAR_IN_OUT
    aMinValue            : TcVnVector4_LREAL;
    aPosition            : TcVnPoint2_DINT;
END_VAR
VAR_INPUT
    ipMask               : ITcVnImage;
    eVectorCompareMethod : ETcVnVectorCompareMethod;
    hrPrev               : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Source image
ipMask	ITcVnImage [▶ 426]	Optional mask to specify which pixel positions are considered (USINT, set parameter to 0 if not required)
eVectorCompareMethod	ETcVnVectorCompareMethod [▶ 234]	Select a vector compare method for multi-channel images
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aMinValue	TcVnVector4_LREAL [▶ 173]	Returns the minimum pixel value
aPosition	TcVnPoint2_DINT [▶ 171]	Returns the first found position of aMinValue (not supported for multi-channel images with ELEMENTWISE)

Return value

HRESULT [▶ 150]

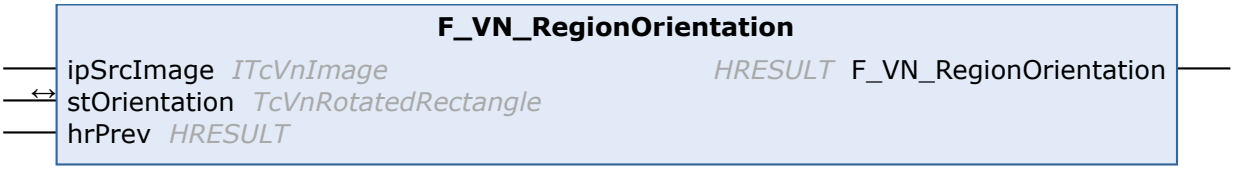
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.26 F_VN_RegionOrientation



Calculate the orientation of a region based on a binary image that contains a non zero value for every pixel in that region.

Syntax

Definition:

```
FUNCTION F_VN_RegionOrientation : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
END_VAR
VAR_IN_OUT
    stOrientation : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel, binary).
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stOrientation	TcVnRotatedRectangle [▶ 258]	Resulting rotated rectangle, containig center of mass, lenghts of axes, and rotation angle of the region in clockwise direction.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.27 F_VN_RegionOrientationExp

Calculate the orientation of a region based on a binary image that contains a non zero value for every pixel in that region. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_RegionOrientationExp : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
END_VAR
VAR_IN_OUT
    stOrientation : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    eMethod       : ETcVnOrientationMethod;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel, binary)
eMethod	ETcVnOrientationMethod [▶ 223]	Method for calculating the orientation.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stOrientation	TcVnRotatedRectangle [▶ 258]	Resulting rotated rectangle, containig center of mass, lenghts of axes, and rotation angle of the region in clockwise direction.

Return value

HRESULT [▶ 150]

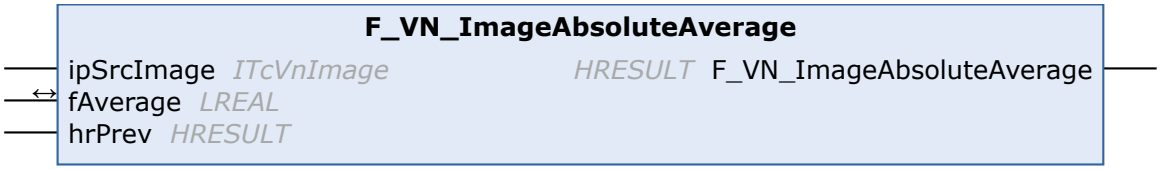
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.28 F_VN_ImageAbsoluteAverage



Computes the average absolute pixel value of an image.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_ImageAbsoluteAverage : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
END_VAR
VAR_IN_OUT
    fAverage   : LREAL;
END_VAR
VAR_INPUT
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fAverage	LREAL	Returns the average absolute pixel value

Return value

HRESULT [▶ 150]

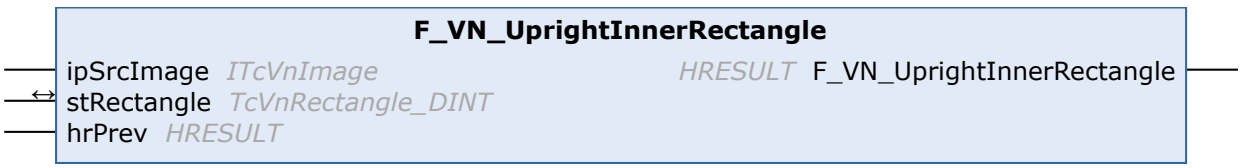
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.29 F_VN_UprightInnerRectangle



Computes the maximum area upright inner rectangle inside an area. If the valid area contains very small invalid areas or the performance is too slow, use the expert variant of this function and manually choose a suitable nDownScaleFactor.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_UprightInnerRectangle : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
END_VAR
VAR_IN_OUT
    stRectangle : TcVnRectangle_DINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, 1 channel, binary, values > 0 mark valid area)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRectangle	TcVnRectangle_DINT [▶ 257]	Returns the maximum area upright inner rectangle

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.14.30 F_VN_UprightInnerRectangleExp

Computes the maximum area upright inner rectangle inside an area. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_UprightInnerRectangleExp : HRESULT
VAR_INPUT
    ipSrcImage       : ITcVnImage;
END_VAR
VAR_IN_OUT
    stRectangle      : TcVnRectangle_DINT;
END_VAR
VAR_INPUT
    nDownScaleFactor : DINT;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, 1 channel, binary, values > 0 mark valid area)
nDownScaleFactor	DINT	To tradeoff performance vs. accuracy, a factor can be provided, to reduce the internally used image resolution. As long as invalid areas enclosed in the valid area are larger than nDownScaleFactor, the result usually still is the optimal one. Use -1 to auto select depending on ipSrcImage resolution, 0 to always use full resolution (slower but always optimal result).
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRectangle	TcVnRectangle_DINT [▶ 257]	Returns the maximum area upright inner rectangle

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15 Image Color and Contrast Processing

Diese Gruppe enthält Funktionen zur Farb- und Kontrastverarbeitung.

Funktionen

Häufigkeitsverteilung, Intensität und Kontrast Anpassung eines Bildes

- F_VN_Clahe(Exp) [▶ 1268]
- F_VN_Histogram(Exp) [▶ 1273]
- F_VN_HistogramEqualization(Exp) [▶ 1274]
- F_VN_NormalizeImage(Exp) [▶ 1280]

- F_VN_NormalizeImageForDisplay [▶ 1282]

Look-Up Table und Color Maps

- F_VN_ApplyColorMap [▶ 1265]
- F_VN_ApplyLut [▶ 1267]
- F_VN_GenerateColorMap [▶ 1270]
- F_VN_GenerateCustomColorMap [▶ 1272]

Color Matching

- F_VN_ReferenceColorSimilarity(Exp) [▶ 1283]
- F_VN_TrainImageColor(Exp) [▶ 1290]
- F_VN_TrainImageColor(Exp)_ITcVnMlModel [▶ 1291]

Weißabgleich

- F_VN_WhiteBalance [▶ 1304]

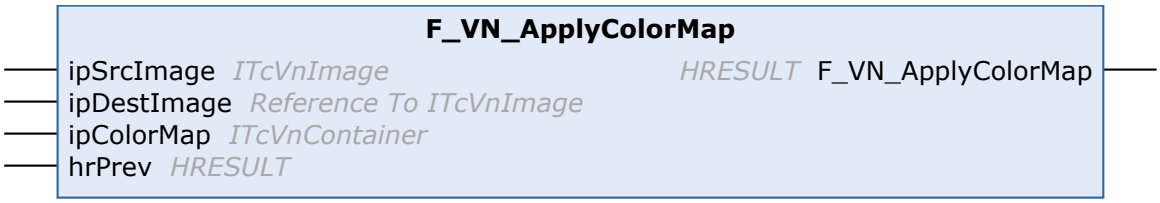
Sonstiges

- F_VN_ConvertColorSpace [▶ 1270]
- F_VN_InvertImageColor(Exp) [▶ 1278]

Beispiele

Image Color and Contrast Processing [▶ 2887]

6.1.4.15.1 F_VN_ApplyColorMap



Apply a color map to a gray-level image.

Syntax

Definition:

```
FUNCTION F_VN_ApplyColorMap : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    ipColorMap  : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (TCVN_ET_USINT or TCVN_ET_UINT, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 426]	Returns the colored image (TCVN_ET_USINT, 3 channel. An appropriate image will be created if required.)
ipColorMap	ITcVnContainer [▶ 385]	Color map to be applied to ipSrcImage (ContainerType_Vector_TcVnVector3_REAL with 256 or 65536 elements, dependent on ipSrcImage type). Can be either custom or created with F_VN_GenerateColorMap.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_ApplyColorMap` wendet eine Farbtabelle auf das Eingangsbild an.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Grauwertbild sein und als Element-Typ `USINT` (8 Bit) oder `UINT` (16 Bit) haben.

Ergebnisbild

Das Ergebnisbild `ipDestImage` gibt das mit der Farbtabelle transformierte, 3-kanalige RGB-Farbbild mit dem Element-Typ `USINT` (8 Bit) zurück.

Farbtabelle

Die Farbtabelle `ipColorMap` definiert, wie das Eingangsbild transformiert wird. Die Farbtabelle ist ein Container, der entweder manuell oder mit einer der beiden folgenden Funktionen erstellt werden kann:

- F_VN_GenerateColorMap [▶ 1270]
- F_VN_GenerateCustomColorMap [▶ 1272]

Der Container muss je nach Element-Typ des Eingangsbilds 256 Elemente (für 8 Bit) oder 65.536 Elemente (für 16 Bit) enthalten und vom Typ `ContainerType_Vector_TcVnVector3_REAL` sein.

Anwendung

Die Anwendung einer typischen Heatmap auf ein 8-Bit-Bild sieht z. B. so aus:

```
hr := F_VN_GenerateColorMap(ipColorMap, TCVN_CMM_HOT, TCVN_CMS_256, hr);
hr := F_VN_ApplyColorMap(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    ipColorMap      :=  ipColorMap,
    hrPrev          :=  hr
);
```

Beispiele

- Darstellung mit Farbtabellen [▶ 2890]

Verwandte Funktionen

- F_VN_GenerateColorMap [▶ 1270]

- F_VN_GenerateCustomColorMap [▶ 1272]
- F_VN_ApplyColorMap [▶ 1265]

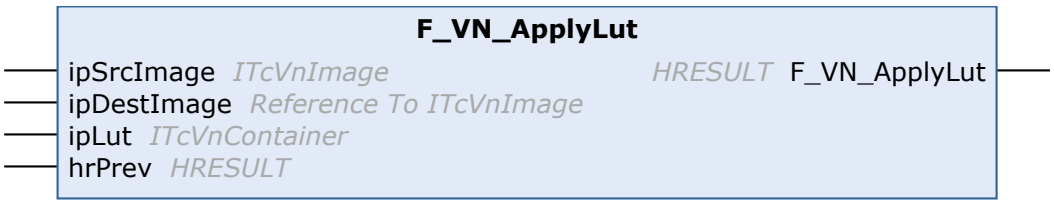
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.2 F_VN_ApplyLut



Apply a lookup table to an image to manipulate its colors.

Syntax

Definition:

```
FUNCTION F_VN_ApplyLut : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    ipLut       : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1-4 channels of 8 or 16 bit types SINT, USINT, INT, UINT)
ipDestImage	Reference To ITcVnImage [▶ 426]	Returns the result image (same size, type and channels as ipSrcImage. An appropriate image will be created if required.)
ipLut	ITcVnContainer [▶ 385]	Lookup table with elements matching ipSrcImage type and either 256 (8 bit images) or 65536 (16 bit images) elements. E.g. ContainerType_Vector_USINT for 1-4 channel USINT image (in this case, the same lookup values are used for each channel) or ContainerType_Vector_TcVnVector3_USINT for a 3 channel USINT image (in this case, each channel uses an individual lookup value).
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.3 F_VN_Clahe

Apply Constrast Limited Adaptive Histogram Equalization. This applies local histogram equalization to individual tiles.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_Clahe : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (TCVN_ET_USINT or TCVN_ET_UINT, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 426]	Returns the resulting image (An appropriate image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.4 F_VN_ClaheExp

Apply Constrast Limited Adaptive Histogram Equalization. This applies local histogram equalization to individual tiles. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_ClaheExp : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    fClipLimit  : LREAL;
    nTilesX     : UDINT;
    nTilesY     : UDINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (TCVN_ET_USINT or TCVN_ET_UINT, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 426]	Returns the resulting image (An appropriate image will be created if required.)
fClipLimit	LREAL	Threshold for contrast limit (set to <= 0 to disable clipping)
nTilesX	UDINT	Number of tiles in x direction
nTilesY	UDINT	Number of tiles in y direction
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

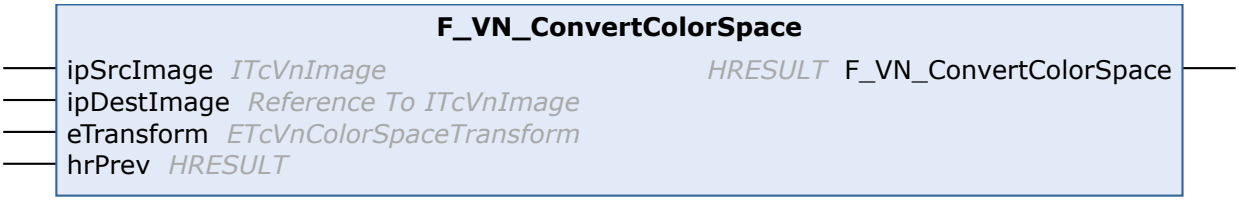
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.5 F_VN_ConvertColorSpace



Convert image from one color space to another.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_ConvertColorSpace : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    eTransform  : ETcVnColorSpaceTransform;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
eTransform	ETcVnColorSpaceTransform [▶ 186]	Transforms to be applied
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

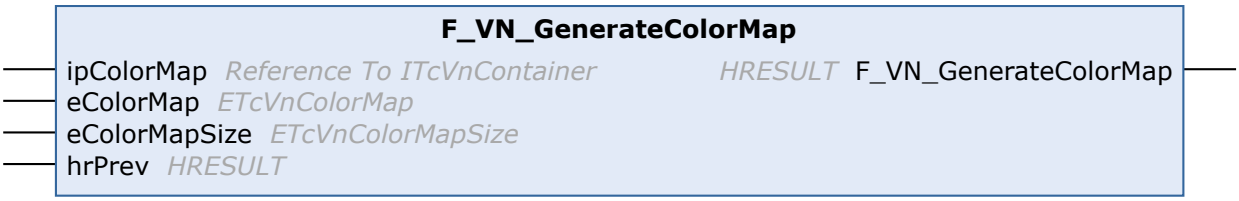
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.6 F_VN_GenerateColorMap



Generates a pre-defined color map.

Syntax

Definition:

```
FUNCTION F_VN_GenerateColorMap : HRESULT
VAR_INPUT
    ipColorMap    : Reference To ITcVnContainer;
    eColorMap     : ETcVnColorMap;
    eColorMapSize : ETcVnColorMapSize;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipColorMap	Reference To ITcVnContainer [▶ 385]	Returns the color map (ContainerType_Vector_TcVnVector3_REAL with 256 or 65536 elements)
eColorMap	ETcVnColorMap [▶ 181]	Selects a color map (similar to GNU Octave/MATLAB types)
eColorMapSize	ETcVnColorMapSize [▶ 186]	Defines how many elements the generated color map should have
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_GenerateColorMap` erstellt eine Farbtabelle, die mit der Funktion F_VN_ApplyColorMap [▶ 1265] auf ein Bild angewandt werden kann.

Parameter

Color Map

Der Parameter `ipColorMap` gibt die erstellte Farbtabelle zurück.

Farbverlauf

Der Farbverlauf `eColorMap` der Farbtabelle wird über das Enum ETcVnColorMap [▶ 181] festgelegt. Wenn Sie manuell einen Farbverlauf festlegen möchten, nutzen Sie die Funktion F_VN_GenerateCustomColorMap [▶ 1272].

Größe der Farbtabelle

Die Größe `eColorMapSize` wird über das Enum ETcVnColorMapSize [▶ 186] festgelegt und definiert, ob die Farbtabelle für 8-Bit-Bilder (`TCVN_CMS_256`) oder für 16-Bit-Bilder (`TCVN_CMS_65536`) erstellt wird.

Anwendung

```
VAR
    ipColorMap      :   ITcVnColorMap;
END_VAR

hr := F_VN_GenerateColorMap(
    ipColorMap      :=  ipColorMap,
    eColorMap       :=  TCVN_CM_HOT,
    eColorMapSize   :=  TCVN_CMS_256,
    hrPrev          :=  hr
);
```

Beispiele

- Darstellung mit Farbtabellen [▶ 2890]

Verwandte Funktionen

- F_VN_GenerateColorMap [▶ 1270]
- F_VN_GenerateCustomColorMap [▶ 1272]
- F_VN_ApplyColorMap [▶ 1265]

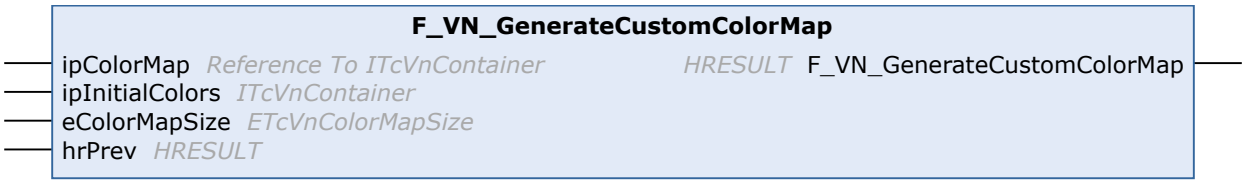
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.7 F_VN_GenerateCustomColorMap



Generate a custom color map by interpolating between user defined colors (equally distributed, linear for each channel).

Syntax

Definition:

```
FUNCTION F_VN_GenerateCustomColorMap : HRESULT
VAR_INPUT
    ipColorMap      : Reference To ITcVnContainer;
    ipInitialColors : ITcVnContainer;
    eColorMapSize   : ETcVnColorMapSize;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipColorMap	Reference To ITcVnContainer [▶ 385]	Returns the color map (ContainerType_Vector_TcVnVector3_REAL with 256 or 65536 elements)
ipInitialColors	ITcVnContainer [▶ 385]	User defined colors (ContainerType_Vector_TcVnVector3_REAL, at least 2 elements)
eColorMapSize	ETcVnColorMapSize [▶ 186]	Defines how many elements the generated color map should have
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Beispiele

- Darstellung mit Farbtabellen [▶ 2890]

Verwandte Funktionen

- F_VN_GenerateColorMap [▶ 1270]
- F_VN_GenerateCustomColorMap [▶ 1272]
- F_VN_ApplyColorMap [▶ 1265]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.8 F_VN_Histogram

Calculate the (multi-channel) histogram of an image.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_Histogram : HRESULT
VAR_INPUT
    ipSrcImage      : ITcVnImage;
    ipDestHistogram : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestHistogram	Reference To ITcVnContainer [▶ 385]	Returns a container with a multi-channel histogram, where every channel is represented as a vector of UDINT (ContainerType_Vector_Vector_UDINT. Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_Histogram` errechnet Kanal-weise das Histogramm des Eingangsbilds.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` darf jedes beliebige Format haben.

Histogramm

Das errechnete mehr-kanalige Histogramm wird als Container `ipDestHistogram` des Typs `ContainerType_Vector_Vector_UDINT` zurückgegeben. Jedes Sub-Element entspricht dem Histogramm eines Kanals. Jeder Behälter des Histogramms wird als `UDINT` repräsentiert, die die Anzahl der zum Behälter gehörenden Pixel angibt.

Expert-Parameter

Die Expert-Variante F_VN_HistogramExp [▶ 1276] enthält zusätzliche Parameter.

Anwendung

Das Ermitteln des am häufigsten vorkommenden Pixelwerts in einem Grauwertbild sieht z.B. so aus:

```
VAR
    ipHistograms        :   ITcVnContainer;
    ipSingleHistogram   :   ITcVnContainer;
    nIndex              :   ULINT;
    ipIterator          :   ITcVnBidirectionalIterator;
END_VAR

hr := F_VN_Histogram(
    ipSrcImage      :=  ipImageIn,
    ipDestHistogram :=  ipHistograms,
    hrPrev          :=  hr);

hr := F_VN_GetAt_ITcVnContainer(ipHistograms, ipSingleHistogram, 0, hr);
hr := F_VN_MaxElement(ipSingleHistogram, ipIterator, nIndex, hr);
// nIndex now contains the most frequent pixel value in ipImageIn.
```

Beispiele

- Umgang mit Histogramm

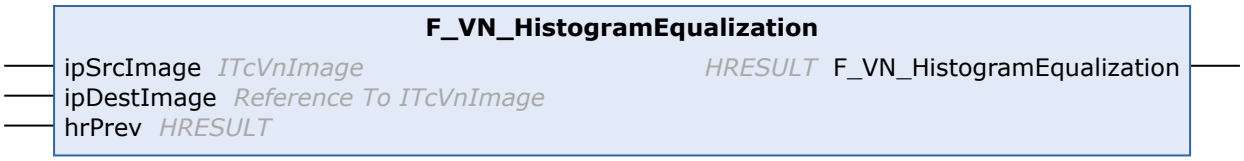
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.9 F_VN_HistogramEqualization



Equalize the histogram of a grayscale or rgb image, which normalizes the brightness and improves the contrast.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_HistogramEqualization : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

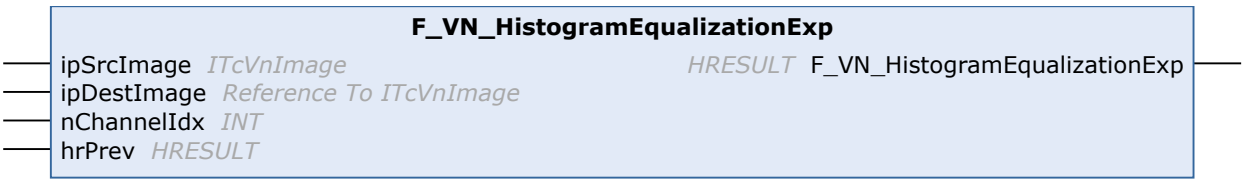
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.10 F_VN_HistogramEqualizationExp



Equalize the histogram of a grayscale or color image, which normalizes the brightness and improves the contrast. (expert function) The channel index that should be equalized has to be specified (-1 expects a RGB image, converts it to YCbCr, equalizes the Y channel and converts the image back to RGB).
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_HistogramEqualizationExp : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    nChannelIdx : INT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
nChannelIdx	INT	Index of the image channel that should be equalized (-1 expects a RGB image, converts it to YCbCr, equalizes the Y channel and converts the image back to RGB)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.11 F_VN_HistogramExp

Calculate the (multi-channel) histogram of an image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_HistogramExp : HRESULT
VAR_INPUT
    ipSrcImage      : ITcVnImage;
    ipDestHistogram : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    nBins           : UDINT;
    fLowerBound     : LREAL;
    fUpperBound     : LREAL;
END_VAR
VAR_INPUT
    ipMask          : ITcVnImage;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestHistogram	Reference To ITcVnContainer [▶ 385]	Returns a container with a multi-channel histogram, where every channel is represented as a vector of UDINT (ContainerType_Vector_Vector_UDINT. Non-zero interface pointers are reused.)
ipMask	ITcVnImage [▶ 426]	Source image mask (TCVN_ET_USINT, 1 channel. Mask is optional, set to 0 if not required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nBins	UDINT	Desired number of bins or 0 to keep the default for the corresponding image format (in) and default number of bins (out)
fLowerBound	LREAL	Lower (inclusive) boundary of the 0-th histogram bin (in), or receive the default if fLowerBound AND fUpperBound are set to 0 (out)
fUpperBound	LREAL	Upper (exclusive) boundary of the last histogram bin nBins-1 (in), or receive the default if fLowerBound AND fUpperBound are set to 0 (out)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_HistogramExp` ist die Expert-Variante von F_VN_Histogram [▶ 1273]. Sie enthält zusätzliche Parameter.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` darf jedes beliebige Format haben.

Histogramm

Das errechnete mehr-kanalige Histogramm wird als Container `ipDestHistogram` des Typs `ContainerType_Vector_Vector_UDINT` zurückgegeben. Jedes Sub-Element entspricht dem Histogramm eines Kanals. Jeder Behälter des Histogramms wird als `UDINT` repräsentiert, die die Anzahl der zum Behälter gehörenden Pixel angibt.

Behälter

Die Anzahl `nBins` definiert, in wie viele Behälter das zu betrachtende Pixelspektrum aufgeteilt werden soll.

Grenzen

Die untere Grenze `fLowerBound` und die obere Grenze `fUpperBound` definieren, welche Pixelwerte bei der Erstellung des Histogramms berücksichtig werden.

Maske

Die Maske `ipMask` definiert, welche Pixel des Eingangsbilds `ipSrcImage` in der Berechnung des Histogramms berücksichtigt werden. Sie wird durch ein 1-kanaliges Bild mit Element-Typ `USINT` beschrieben, das dieselbe Größe wie das Eingangsbild `ipSrcImage` besitzt. Alle Pixel, die im Maskenbild einen Wert `<> 0` haben, werden bei der Berechnung des Histogramms berücksichtigt.

Anwendung

Die Berechnung eines Histogramms mit 10 Behältern für den Wertebereich 100-200 mit einer kreisförmigen Maske sieht z. B. so aus:

```
VAR
    ipImageMask :   ITcVnImage;
END_VAR

// Create mask image with a circle at image center.
hr := F_VN_CopyImage(ipImageIn, ipImageMask, hr);
hr := F_VN_SetPixels(ipImageMask, aBlack, hr);
hr := F_VN_DrawCircle(400, 300, 300, ipImageMask, aWhite, -1, hr);

hr := F_VN_HistogramExp(
    ipSrcImage      :=  ipImageIn,
    ipDestHistogram :=  ipHistograms,
    nBins           :=  10,
    fLowerBound     :=  100,
    fUpperBound     :=  201,
    ipMask          :=  ipImageMask,
    hrPrev          :=  hr
);
```

Beispiele

- Umgang mit Histogramm

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.12 F_VN_InvertImageColor

Invert the color of an image. For signed integer or negative floating point values, only the signs will be switched. If the image contains only positive floating point values, each pixel value is subtracted from the maximum available pixel value (or 1.0, whatever is higher).

Syntax

Definition:

```
FUNCTION F_VN_InvertImageColor : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

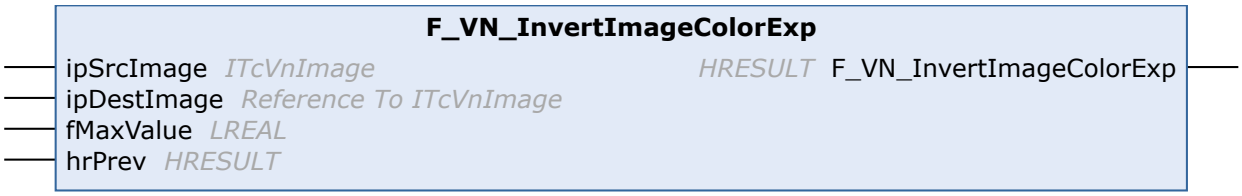
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.13 F_VN_InvertImageColorExp



Invert the color of an image. (expert function) For signed integer or negative floating point values, only the signs will be switched. If the image contains only positive floating point values, each pixel value is subtracted from the maximum available pixel value (or 1.0, whatever is higher).

Syntax

Definition:

```
FUNCTION F_VN_InvertImageColorExp : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    fMaxValue   : LREAL;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate image will be created if required.)
fMaxValue	LREAL	Maximum pixel value (e.g. if a 16 bit image contains 12 bit values). -1 means the default values are used.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

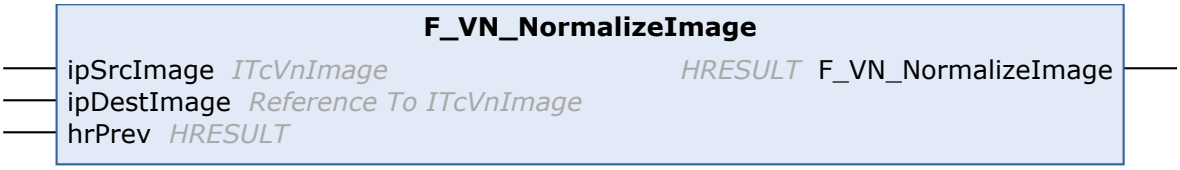
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.14 F_VN_NormalizeImage



Normalize an image regarding its value range (e.g. stretch pixel values [50..150] to full range [0..255]).

Syntax

Definition:

```
FUNCTION F_VN_NormalizeImage : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

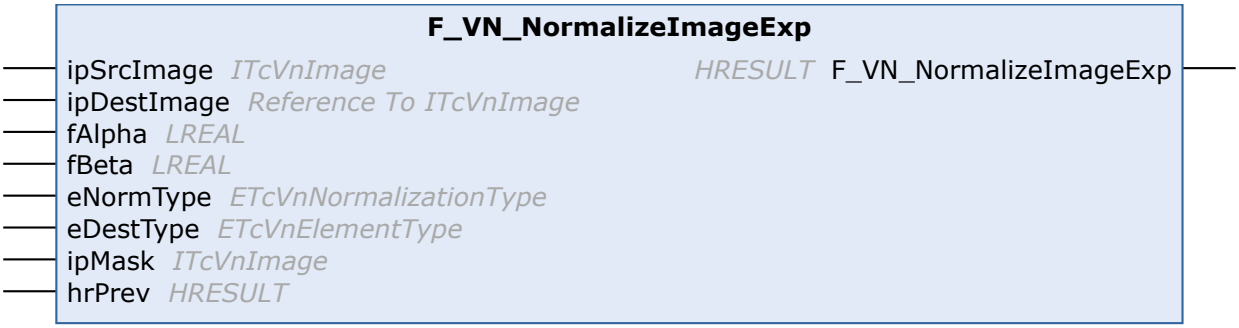
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.15 F_VN_NormalizeImageExp



Normalize an image regarding its value range (e.g. stretch pixel values [50..150] to full range [0..255]) or scale the values regarding a specific normalization (e.g. L2-norm). (expert function)

Syntax

Definition:

```
FUNCTION F_VN_NormalizeImageExp : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    fAlpha      : LREAL;
    fBeta       : LREAL;
    eNormType   : ETcVnNormalizationType;
    eDestType   : ETcVnElementType;
    ipMask      : ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
fAlpha	LREAL	Lower range boundary (in case of normalizing the value range) or value to normalize to (||ipDestImage|| = fAlpha)
fBeta	LREAL	Upper range boundary (in case of normalizing the value range)
eNormType	ETcVnNormalizationType [▶ 221]	Normalization type (only INF, L1, L2 or MINMAX)
eDestType	ETcVnElementType [▶ 210]	Destination image depth (usually SAME_AS_SOURCE)
ipMask	ITcVnImage [▶ 426]	Mask to restrict the normalization to specific pixel positions (set 0 to normalize the whole image)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

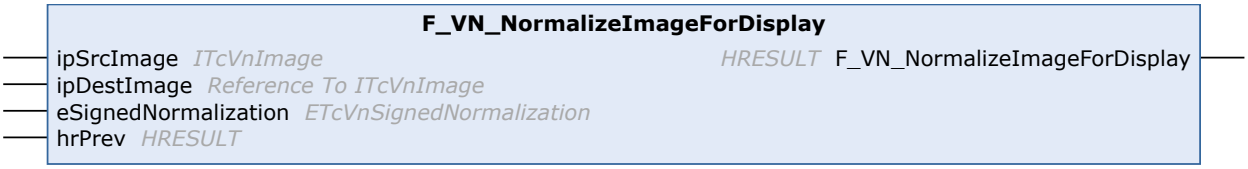
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.16 F_VN_NormalizeImageForDisplay



Normalize an image for display, i.e. scale it to the full value range of the underlying data type (-1 to 1 for floating point).
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_NormalizeImageForDisplay : HRESULT
VAR_INPUT
    ipSrcImage           : ITcVnImage;
    ipDestImage          : Reference To ITcVnImage;
    eSignedNormalization : ETcVnSignedNormalization;
    hrPrev               : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
eSignedNormalization	ETcVnSignedNormalization [▶ 228]	Option for normalizing signed values
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

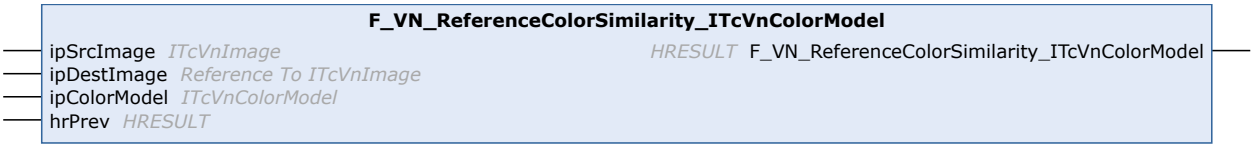
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.17 F_VN_ReferenceColorSimilarity_ITcVnColorModel



Computes the similarity to a reference color model for each pixel in the source image.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_ReferenceColorSimilarity_ITcVnColorModel : HRESULT
VAR_INPUT
    ipSrcImage   : ITcVnImage;
    ipDestImage  : Reference To ITcVnImage;
    ipColorModel : ITcVnColorModel;
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	RGB source image (USINT, 3 channels)
ipDestImage	Reference To ITcVnImage [▶ 426]	Returns the similarity to ipColorModel for each pixel in ipSrcImage (USINT, 1 channel. An appropriate destination image will be created if required.)
ipColorModel	ITcVnColorModel [▶ 433]	Color model
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Es gibt drei Möglichkeiten die Referenzfarbe an die entsprechende Funktion zu übergeben:

- Als trainiertes Farbmodell vom Typ `ITcVnColorModel`
- Als trainiertes Farbmodell vom Typ `ITcVnMlModel`
- Direkt mit Werten in einem `TcVnVector3_LREAL`, siehe Beispiel

Zum Trainieren eines Farbmodells muss eine entsprechende F_VN_TrainImageColor [▶ 1290] Funktion verwendet werden. Die beiden Farbmodelle unterscheiden sich nur im Datentyp, funktionell gibt es keine Unterschiede. Der `ITcVnMlModel` ist der allgemeine Datentyp für ML-Modelle für den es auch Funktionsbausteine zum Laden und Speichern gibt. Daher sollte dieser Datentyp und die entsprechenden Funktionen verwendet werden, wenn das Trainierte Farbmodell auf der Festplatte gespeichert werden soll.

Beispiele

- Color Similarity mit RGB-Referenz-Farbe [▶ 2887]

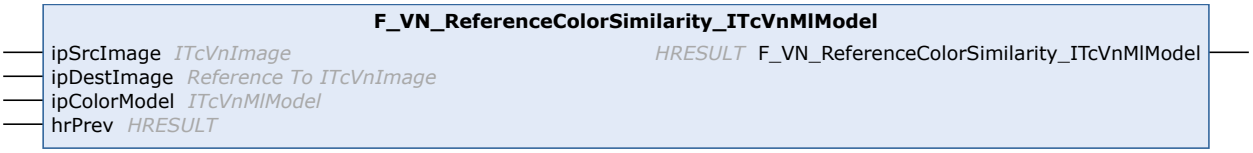
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.18 F_VN_ReferenceColorSimilarity_ITcVnMlModel



Computes the similarity to a reference color model for each pixel in the source image.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_ReferenceColorSimilarity_ITcVnMlModel : HRESULT
VAR_INPUT
    ipSrcImage   : ITcVnImage;
    ipDestImage  : Reference To ITcVnImage;
    ipColorModel : ITcVnMlModel;
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	RGB source image (USINT, 3 channels)
ipDestImage	Reference To ITcVnImage [▶ 426]	Returns the similarity to ipColorModel for each pixel in ipSrcImage (USINT, 1 channel. An appropriate destination image will be created if required.)
ipColorModel	ITcVnMlModel [▶ 435]	Color model
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Es gibt drei Möglichkeiten die Referenzfarbe an die entsprechende Funktion zu übergeben:

- Als trainiertes Farbmodell vom Typ `ITcVnColorModel`
- Als trainiertes Farbmodell vom Typ `ITcVnMlModel`
- Direkt mit Werten in einem `TcVnVector3_LREAL`, siehe Beispiel

Zum Trainieren eines Farbmodells muss eine entsprechende F_VN_TrainImageColor [▶ 1290] Funktion verwendet werden. Die beiden Farbmodelle unterscheiden sich nur im Datentyp, funktionell gibt es keine Unterschiede. Der `ITcVnMlModel` ist der allgemeine Datentyp für ML-Modelle für den es auch Funktionsbausteine zum Laden und Speichern gibt. Daher sollte dieser Datentyp und die entsprechenden Funktionen verwendet werden, wenn das Trainierte Farbmodell auf der Festplatte gespeichert werden soll.

Beispiele

- Color Similarity mit RGB-Referenz-Farbe [▶ 2887]

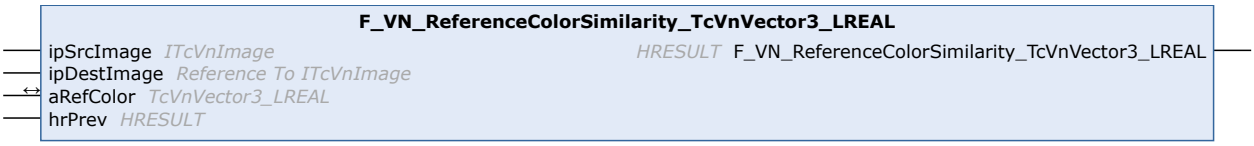
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.19 F_VN_ReferenceColorSimilarity_TcVnVector3_LREAL



Computes the similarity to a reference color for each pixel in the source image.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_ReferenceColorSimilarity_TcVnVector3_LREAL : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    aRefColor   : TcVnVector3_LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	RGB source image (USINT, 3 channels)
ipDestImage	Reference To ITcVnImage [▶ 426]	Returns the similarity to aRefColor for each pixel in ipSrcImage (USINT, 1 channel. An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aRefColor	TcVnVector3_LREAL [▶ 173]	Reference color (RGB, [0..255])

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.20 F_VN_ReferenceColorSimilarityExp_ITcVnColorModel

Computes the similarity to a reference color model for each pixel in the source image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_ReferenceColorSimilarityExp_ITcVnColorModel : HRESULT
VAR_INPUT
    ipSrcImage       : ITcVnImage;
    ipDestImage      : Reference To ITcVnImage;
    ipColorModel     : ITcVnColorModel;
    fVariance        : REAL;
    fLuminanceWeight : REAL;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	RGB source image (USINT, 3 channels)
ipDestImage	Reference To ITcVnImage [▶ 426]	Returns the similarity to ipColorModel for each pixel in ipSrcImage (USINT, 1 channel. An appropriate destination image will be created if required.)
ipColorModel	ITcVnColorModel [▶ 433]	Color model
fVariance	REAL	Allowed color variance (0.1 - 0.3 might be a good start to try)
fLuminanceWeight	REAL	Weight the impact of the luminance ([0..1], e.g. set to 0 to be more resistant to unequal illumination, but might be required to differentiate between some colors. Ignored if ipColorModel is of type RGB.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

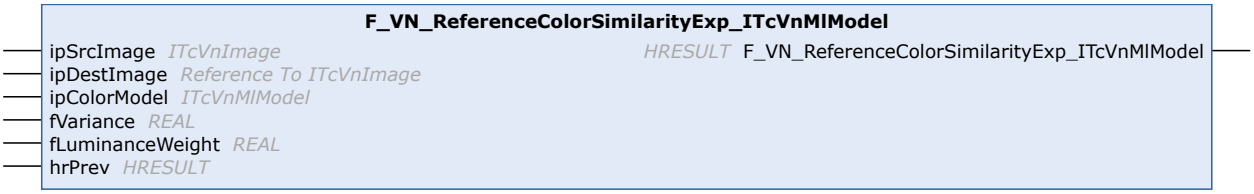
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.21 F_VN_ReferenceColorSimilarityExp_ITcVnMlModel



Computes the similarity to a reference color model for each pixel in the source image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_ReferenceColorSimilarityExp_ITcVnMlModel : HRESULT
VAR_INPUT
    ipSrcImage       : ITcVnImage;
    ipDestImage      : Reference To ITcVnImage;
    ipColorModel     : ITcVnMlModel;
    fVariance        : REAL;
    fLuminanceWeight : REAL;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	RGB source image (USINT, 3 channels)
ipDestImage	Reference To ITcVnImage [▶ 426]	Returns the similarity to ipColorModel for each pixel in ipSrcImage (USINT, 1 channel. An appropriate destination image will be created if required.)
ipColorModel	ITcVnMlModel [▶ 435]	Color model
fVariance	REAL	Allowed color variance (0.1 - 0.3 might be a good start to try)
fLuminanceWeight	REAL	Weight the impact of the luminance ([0..1], e.g. set to 0 to be more resistant to unequal illumination, but might be required to differentiate between some colors. Ignored if ipColorModel is of type RGB.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Es gibt drei Möglichkeiten die Referenzfarbe an die entsprechende Funktion zu übergeben:

- Als trainiertes Farbmodell vom Typ `ITcVnColorModel`
- Als trainiertes Farbmodell vom Typ `ITcVnMlModel`
- Direkt mit Werten in einem `TcVnVector3_LREAL`, siehe Beispiel

Zum Trainieren eines Farbmodells muss eine entsprechende F_VN_TrainImageColor [▶ 1290] Funktion verwendet werden. Die beiden Farbmodelle unterscheiden sich nur im Datentyp, funktionell gibt es keine Unterschiede. Der `ITcVnMlModel` ist der allgemeine Datentyp für ML-Modelle für den es auch Funktionsbausteine zum Laden und Speichern gibt. Daher sollte dieser Datentyp und die entsprechenden Funktionen verwendet werden, wenn das Trainierte Farbmodell auf der Festplatte gespeichert werden soll.

Beispiele

- Color Similarity mit RGB-Referenz-Farbe [▶ 2887]

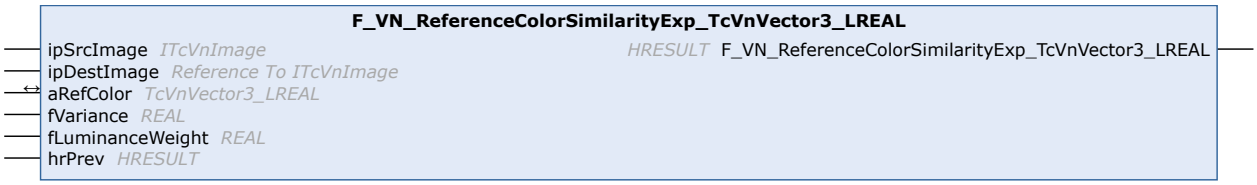
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.22 F_VN_ReferenceColorSimilarityExp_TcVnVector3_LREAL



Computes the similarity to a reference color for each pixel in the source image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_ReferenceColorSimilarityExp_TcVnVector3_LREAL : HRESULT
VAR_INPUT
    ipSrcImage       : ITcVnImage;
    ipDestImage      : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    aRefColor        : TcVnVector3_LREAL;
END_VAR
VAR_INPUT
    fVariance        : REAL;
    fLuminanceWeight : REAL;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	RGB source image (USINT, 3 channels)
ipDestImage	Reference To ITcVnImage [▶ 426]	Returns the similarity to aRefColor for each pixel in ipSrcImage (USINT, 1 channel. An appropriate destination image will be created if required.)
fVariance	REAL	Allowed color variance (0.1 - 0.3 might be a good start to try)
fLuminanceWeight	REAL	Weight the impact of the luminance ([0..1], e.g. set to 0 to be more resistant to unequal illumination, but might be required to differentiate between some colors)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aRefColor	TcVnVector3_LREAL [▶ 173]	Reference color (RGB, [0..255])

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Es gibt drei Möglichkeiten die Referenzfarbe an die entsprechende Funktion zu übergeben:

- Als trainiertes Farbmodell vom Typ `ITcVnColorModel`
- Als trainiertes Farbmodell vom Typ `ITcVnMlModel`
- Direkt mit Werten in einem `TcVnVector3_LREAL`, siehe Beispiel

Zum Trainieren eines Farbmodells muss eine entsprechende F_VN_TrainImageColor [▶ 1290] Funktion verwendet werden. Die beiden Farbmodelle unterscheiden sich nur im Datentyp, funktionell gibt es keine Unterschiede. Der `ITcVnMlModel` ist der allgemeine Datentyp für ML-Modelle für den es auch Funktionsbausteine zum Laden und Speichern gibt. Daher sollte dieser Datentyp und die entsprechenden Funktionen verwendet werden, wenn das Trainierte Farbmodell auf der Festplatte gespeichert werden soll.

Beispiele

- Color Similarity mit RGB-Referenz-Farbe [▶ 2887]

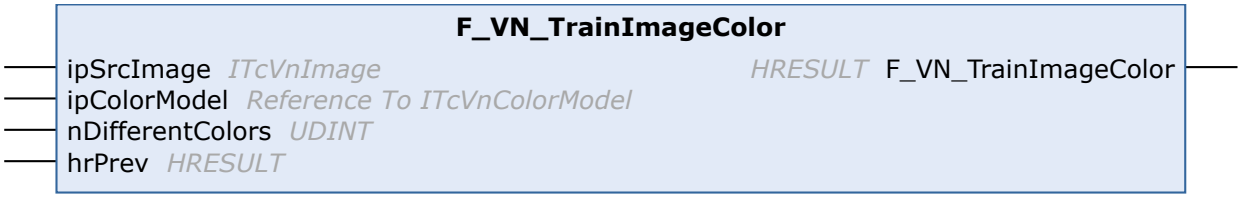
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.23 F_VN_TrainImageColor



Create a new color model, describing the image color.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_TrainImageColor : HRESULT
VAR_INPUT
    ipSrcImage       : ITcVnImage;
    ipColorModel     : Reference To ITcVnColorModel;
    nDifferentColors : UDINT;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (3 channels (RGB) of type USINT)
ipColorModel	Reference To ITcVnColorModel [▶ 433]	Returns the color model
nDifferentColors	UDINT	Number of different colors to distinguish
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_TrainImageColor` trainiert ein Farbmodell basierend auf einem Referenzbild. Wenn das trainierte Farbmodell auf der Festplatte gespeichert werden soll, muss die F_VN_TrainImageColor_ITcVnMlModel [▶ 1291] mit dem Typ `ITcVnMlModel` verwendet werden.

Parameter

Referenzbild

Das Referenzbild `ipSrcImage` muss 3 Kanäle im RGB-Format und den Element-Typ `USINT` (8 Bit) haben.

Farbmodell

Der Parameter `ipColorModel` gibt das trainierte Farbmodell zurück. Dieses ist ein Interface Pointer vom Typ ITcVnColorModel [▶ 433].

Anzahl zu unterscheidender Farben

Die Anzahl `nDifferentColors` definiert die Anzahl an zu trainierenden Farben. Achten Sie darauf, der Bildhintergrund z. B. auch eine eigene Farbe sein kann.

Expert-Parameter

Weitere Parameter finden Sie in der Expert-Variante F_VN_TrainImageColorExp [▶ 1293].

Anwendung

Das Trainieren eines Farbmodells für ein Referenzbild mit 4 verschiedenfarbigen Objekten sowie dunklem Hintergrund sieht z.B. so aus:

```
VAR
    ipColorModel    :   ITcVnColorModel;
END_VAR

hr := F_VN_TrainImageColor(
            ipSrcImage      := ipImageRef,
            ipColorModel    := ipColorModel,
            nDifferentColors:= 5, // 4 colors + 1 background = 5
            hrPrev          := hr);
```

Verwandte Funktionen

Es gibt drei Möglichkeiten die Referenzfarbe an die entsprechende `F_VN_ReferenceColorSimilarity` Funktion zu übergeben:

- Als trainiertes Farbmodell vom Typ `ITcVnColorModel`
- Als trainiertes Farbmodell vom Typ `ITcVnMlModel`
- Direkt mit Werten in einem `TcVnVector3_LREAL`

Für die Farbmodelle gibt es entsprechende Funktionen zum Trainieren und für alle drei Datentypen jeweils eine Funktion zur Ausführung:

- F_VN_TrainImageColor [▶ 1290] zum Trainieren eines Farbmodells
- F_VN_ReferenceColorSimilarity [▶ 1283] zur Segmentierung mittels Farbmodell

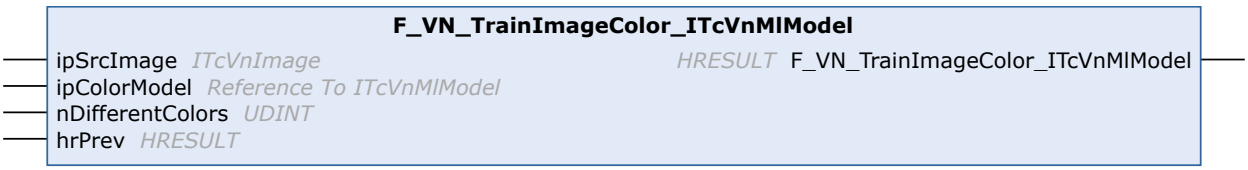
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.24 F_VN_TrainImageColor_ITcVnMlModel



Create a new color model, describing the image color.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_TrainImageColor_ITcVnMlModel : HRESULT
VAR_INPUT
    ipSrcImage       : ITcVnImage;
    ipColorModel     : Reference To ITcVnMlModel;
```

```
    nDifferentColors : UDINT;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (3 channels (RGB) of type USINT)
ipColorModel	Reference To ITcVnMlModel [▶ 435]	Returns the color model
nDifferentColors	UDINT	Number of different colors to distinguish
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_TrainImageColor_ITcVnMlModel` trainiert ein Farbmodell basierend auf einem Referenzbild. Das Farbmodel kann mit den entsprechenden Funktionsbausteinen auf der Festplatte gespeichert und geladen werden.

Parameter

Referenzbild

Das Referenzbild `ipSrcImage` muss 3 Kanäle im RGB-Format und den Element-Typ `USINT` (8 Bit) haben.

Farbmodell

Der Parameter `ipColorModel` gibt das trainierte Farbmodell zurück. Dieses ist ein Interface Pointer vom Typ ITcVnMlModel [▶ 435].

Anzahl zu unterscheidender Farben

Die Anzahl `nDifferentColors` definiert die Anzahl an zu trainierenden Farben. Achten Sie darauf, der Bildhintergrund z. B. auch eine eigene Farbe sein kann.

Expert-Parameter

Weitere Parameter finden Sie in der Expert-Variante F_VN_TrainImageColorExp_ITcVnMlModel [▶ 1295].

Anwendung

Das Trainieren eines Farbmodells für ein Referenzbild mit 4 verschiedenfarbigen Objekten sowie dunklem Hintergrund sieht z.B. so aus:

```
VAR
    ipColorModel    :   ITcVnMlModel;
END_VAR

hr := F_VN_TrainImageColor_ITcVnMlModel(
            ipSrcImage      := ipImageRef,
            ipColorModel    := ipColorModel,
            nDifferentColors:= 5, // 4 colors + 1 background = 5
            hrPrev          := hr);
```

Verwandte Funktionen

Es gibt drei Möglichkeiten die Referenzfarbe an die entsprechende `F_VN_ReferenceColorSimilarity` Funktion zu übergeben:

- Als trainiertes Farbmodell vom Typ `ITcVnColorModel`
- Als trainiertes Farbmodell vom Typ `ITcVnMlModel`
- Direkt mit Werten in einem `TcVnVector3_LREAL`

Für die Farbmodelle gibt es entsprechende Funktionen zum Trainieren und für alle drei Datentypen jeweils eine Funktion zur Ausführung:

- F_VN_TrainImageColor [▶ 1290] zum Trainieren eines Farbmodells
- F_VN_ReferenceColorSimilarity [▶ 1283] zur Segmentierung mittels Farbmodell

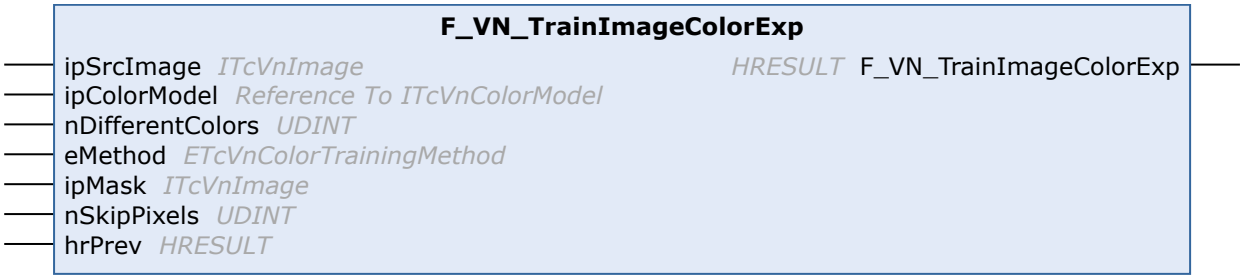
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.25 F_VN_TrainImageColorExp



Create a new color model, describing the image color. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_TrainImageColorExp : HRESULT
VAR_INPUT
    ipSrcImage       : ITcVnImage;
    ipColorModel     : Reference To ITcVnColorModel;
    nDifferentColors : UDINT;
    eMethod          : ETcVnColorTrainingMethod;
    ipMask           : ITcVnImage;
    nSkipPixels      : UDINT;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (3 channels (RGB) of type USINT)
ipColorModel	Reference To ITcVnColorModel [▶ 433]	Returns the color model
nDifferentColors	UDINT	Number of different colors to distinguish
eMethod	ETcVnColorTrainingMethod [▶ 198]	Color training method
ipMask	ITcVnImage [▶ 426]	Optional image mask (1 channel of type USINT, set to 0 if not required)
nSkipPixels	UDINT	Number of pixels to skip between each evaluated color sample (to achieve a better performance). 0 takes every pixel into account and tends to be more accurate.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_TrainImageColorExp` ist die Expert-Variante von F_VN_TrainImageColor [▶ 1290]. Sie enthält weitere Parameter. Wenn das Trainierte Farbmodell auf der Festplatte gespeichert werden soll, muss die F_VN_TrainImageColorExp_ITcVnMlModel [▶ 1295] mit dem Typ `ITcVnMlModel` verwendet werden.

Parameter

Referenzbild

Das Referenzbild `ipSrcImage` muss 3 Kanäle im RGB-Format und den Element-Typ `USINT` (8 Bit) haben.

Farbmodell

Der Parameter `ipColorModel` gibt das trainierte Farbmodell zurück. Dieses ist ein Interface Pointer vom Typ ITcVnColorModel [▶ 433].

Anzahl zu unterscheidender Farben

Die Anzahl `nDifferentColors` definiert die Anzahl an zu trainierenden Farben. Achten Sie darauf, der Bildhintergrund z. B. auch eine eigene Farbe sein kann.

Expert-Parameter

Zusätzliche Parameter finden Sie in einer weiteren Expert-Variante F_VN_TrainImageColorExp2 [▶ 1298].

Trainingsmethoden

Bei der Auswahl der Trainingsmethode stehen die Farbräume LAB und RGB zur Verfügung.

Maske

Die Maske `ipMask` definiert, welche Pixel des Eingangsbilds `ipSrcImage` in der Berechnung berücksichtigt werden. Sie wird durch ein 1-kanaliges Bild mit Element-Typ USINT beschrieben, dass dieselbe Größe wie das Eingangsbild `ipSrcImage` besitzt. Alle Pixel, die im Maskenbild einen Wert <> 0 haben, werden bei der Berechnung der Cluster berücksichtigt.

Pixel-Lücke

Zur Performance Optimierung kann eine Anzahl von Pixeln übersprungen werden, so dass weniger Pixel bei der Aufteilung in die einzelnen Cluster berücksichtigt werden müssen.

Anwendung

```
VAR
    ipColorModel    :   ITcVnColorModel;
END_VAR

hr := F_VN_TrainImageColor(
            ipSrcImage      :=  ipImageRef,
            ipColorModel    :=  ipColorModel,
            nDifferentColors:=  5, // 4 colors + 1 background = 5
            eMethod         :=  TCVN_CTM_LAB,
            ipMask          :=  0,
            nSkipPixels     :=  3,
            hrPrev          :=  hr);
```

Verwandte Funktionen

Es gibt drei Möglichkeiten die Referenzfarbe an die entsprechende `F_VN_ReferenceColorSimilarity` Funktion zu übergeben:

- Als trainiertes Farbmodell vom Typ `ITcVnColorModel`
- Als trainiertes Farbmodell vom Typ `ITcVnMlModel`
- Direkt mit Werten in einem `TcVnVector3_LREAL`

Für die Farbmodelle gibt es entsprechende Funktionen zum Trainieren und für alle drei Datentypen jeweils eine Funktion zur Ausführung:

- F_VN_TrainImageColor [▶ 1290] zum Trainieren eines Farbmodells
- F_VN_ReferenceColorSimilarity [▶ 1283] zur Segmentierung mittels Farbmodell

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.26 F_VN_TrainImageColorExp_ITcVnMlModel

Create a new color model, describing the image color. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_TrainImageColorExp_ITcVnMlModel : HRESULT
VAR_INPUT
    ipSrcImage       : ITcVnImage;
    ipColorModel     : Reference To ITcVnMlModel;
    nDifferentColors : UDINT;
    eMethod          : ETcVnColorTrainingMethod;
    ipMask           : ITcVnImage;
    nSkipPixels      : UDINT;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (3 channels (RGB) of type USINT)
ipColorModel	Reference To ITcVnMlModel [▶ 435]	Returns the color model
nDifferentColors	UDINT	Number of different colors to distinguish
eMethod	ETcVnColorTrainingMethod [▶ 198]	Color training method
ipMask	ITcVnImage [▶ 426]	Optional image mask (1 channel of type USINT, set to 0 if not required)
nSkipPixels	UDINT	Number of pixels to skip between each evaluated color sample (to achieve a better performance). 0 takes every pixel into account and tends to be more accurate.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_TrainImageColorExp_ITcVnMlModel` ist die Expert-Variante von F_VN_TrainImageColor_ITcVnMlModel [▶ 1291]. Sie enthält weitere Parameter. Das Farbmodel kann mit den entsprechenden Funktionsbausteinen auf der Festplatte gespeichert und geladen werden.

Parameter

Referenzbild

Das Referenzbild `ipSrcImage` muss 3 Kanäle im RGB-Format und den Element-Typ `USINT` (8 Bit) haben.

Farbmodell

Der Parameter `ipColorModel` gibt das trainierte Farbmodell zurück. Dieses ist ein Interface Pointer vom Typ ITcVnMlModel [▶ 435].

Anzahl zu unterscheidender Farben

Die Anzahl `nDifferentColors` definiert die Anzahl an zu trainierenden Farben. Achten Sie darauf, der Bildhintergrund z. B. auch eine eigene Farbe sein kann.

Expert-Parameter

Zusätzliche Parameter finden Sie in einer weiteren Expert-Variante F_VN_TrainImageColorExp2_ITcVnMlModel [▶ 1301].

Trainingsmethoden

Bei der Auswahl der Trainingsmethode stehen die Farbräume LAB und RGB zur Verfügung.

Maske

Die Maske `ipMask` definiert, welche Pixel des Eingangsbilds `ipSrcImage` in der Berechnung berücksichtigt werden. Sie wird durch ein 1-kanaliges Bild mit Element-Typ USINT beschrieben, dass dieselbe Größe wie das Eingangsbild `ipSrcImage` besitzt. Alle Pixel, die im Maskenbild einen Wert <> 0 haben, werden bei der Berechnung der Cluster berücksichtigt.

Pixel-Lücke

Zur Performance Optimierung kann eine Anzahl von Pixeln übersprungen werden, so dass weniger Pixel bei der Aufteilung in die einzelnen Cluster berücksichtigt werden müssen.

Anwendung

```
VAR
    ipColorModel    :   ITcVnMlModel;
END_VAR

hr := F_VN_TrainImageColorExp_ITcVnMlModel (
            ipSrcImage      := ipImageRef,
            ipColorModel    := ipColorModel,
            nDifferentColors:= 5, // 4 colors + 1 background = 5
            eMethod         := TCVN_CTM_LAB,
            ipMask          := 0,
            nSkipPixels     := 3,
            hrPrev          := hr);
```

Verwandte Funktionen

Es gibt drei Möglichkeiten die Referenzfarbe an die entsprechende `F_VN_ReferenceColorSimilarity` Funktion zu übergeben:

- Als trainiertes Farbmodell vom Typ `ITcVnColorModel`
- Als trainiertes Farbmodell vom Typ `ITcVnMlModel`
- Direkt mit Werten in einem `TcVnVector3_LREAL`

Für die Farbmodelle gibt es entsprechende Funktionen zum Trainieren und für alle drei Datentypen jeweils eine Funktion zur Ausführung:

- F_VN_TrainImageColor [▶ 1290] zum Trainieren eines Farbmodells
- F_VN_ReferenceColorSimilarity [▶ 1283] zur Segmentierung mittels Farbmodell

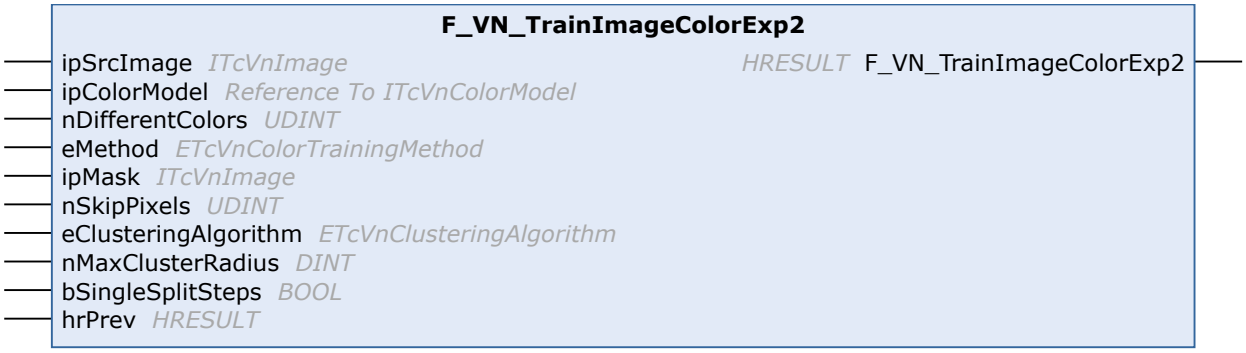
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.27 F_VN_TrainImageColorExp2



Create a new color model, describing the image color. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_TrainImageColorExp2 : HRESULT
VAR_INPUT
    ipSrcImage           : ITcVnImage;
    ipColorModel         : Reference To ITcVnColorModel;
    nDifferentColors     : UDINT;
    eMethod              : ETcVnColorTrainingMethod;
    ipMask               : ITcVnImage;
    nSkipPixels          : UDINT;
    eClusteringAlgorithm : ETcVnClusteringAlgorithm;
    nMaxClusterRadius    : DINT;
    bSingleSplitSteps    : BOOL;
    hrPrev               : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (3 channels (RGB) of type USINT)
ipColorModel	Reference To ITcVnColorModel [▶ 433]	Returns the color model
nDifferentColors	UDINT	Maximum number of different colors to distinguish (if LBG is used as a clustering algorithm, the result might have less different colors, depending on nMaxClusterRadius)
eMethod	ETcVnColorTrainingMethod [▶ 198]	Color training method
ipMask	ITcVnImage [▶ 426]	Optional image mask (1 channel of type USINT, set to 0 if not required)
nSkipPixels	UDINT	Number of pixels to skip between each evaluated color sample (to achieve a better performance). 0 takes every pixel into account and tends to be more accurate.
eClusteringAlgorithm	ETcVnClusteringAlgorithm [▶ 180]	Clustering algorithm
nMaxClusterRadius	DINT	Only used for the LBG clustering algorithm. Maximum allowed radius (> 0) of a single cluster, i.e. clusters with a higher radius will be split into smaller ones, until a global number of nDifferentColors is reached.
bSingleSplitSteps	BOOL	Only used for the LBG clustering algorithm. If true, the global optimization is always run after a single cluster has been split. If false, several clusters are split within the same step before applying the global optimization. Applying the global optimization less often is faster, but can lead to less optimal results, especially having 2 nearby clusters that could be represented by 1)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_TrainImageColorExp2` ist die erweiterte Expert-Variante von F_VN_TrainImageColorExp [▶ 1293]. Sie enthält weitere Parameter. Wenn das Trainierte Farbmodell auf der Festplatte gespeichert werden soll, muss die F_VN_TrainImageColorExp2_ITcVnMlModel [▶ 1301] mit dem Typ `ITcVnMlModel` verwendet werden.

Parameter

Referenzbild

Das Referenzbild `ipSrcImage` muss 3 Kanäle im RGB-Format und den Element-Typ `USINT` (8 Bit) haben.

Farbmodell

Der Parameter `ipColorModel` gibt das trainierte Farbmodell zurück. Dieses ist ein Interface Pointer vom Typ ITcVnColorModel [▶ 433].

Anzahl zu unterscheidender Farben

Die Anzahl `nDifferentColors` definiert die Anzahl an zu trainierenden Farben. Achten Sie darauf, der Bildhintergrund z. B. auch eine eigene Farbe sein kann.

Clustering-Algorithmus

Bei der Auswahl des Clustering-Algorithmus stehen KMeans++ (feste Anzahl von Clustern) und LBG (dynamische Anzahl von Clustern) zur Verfügung.

Maximaler Cluster-Radius

Wird nur beim LBG-Clusteralgorithmus verwendet. Damit kann die Aufteilung der einzelnen Cluster beeinflusst werden.

Globale Optimierung

Wird nur beim LBG-Clusteralgorithmus verwendet. Damit kann zwischen kürzerer Ausführungszeit (`bSingleSplitSteps = FALSE`) und verbesserten Ergebnissen (`bSingleSplitSteps = TRUE`) gewählt werden.

Eine seltenere Anwendung der globalen Optimierung ist schneller, kann aber zu weniger optimalen Ergebnissen führen.

Anwendung

```
VAR
    ipColorModel    :   ITcVnColorModel;
END_VAR

hr := F_VN_TrainImageColorExp2(
            ipSrcImage          := ipImageRef,
            ipColorModel        := ipColorModel,
            nDifferentColors    := 5, // 4 colors + 1 background = 5
            eMethod             := TCVN_CTM_LAB,
            ipMask              := 0,
            nSkipPixels         := 3,
            eClusteringAlgorithm:= TCVN_CA_KMEANSPP,
            nMaxClusterRadius   := 0,
            bSingleSplitSteps   := FALSE,
            hrPrev              := hr);
```

Verwandte Funktionen

Es gibt drei Möglichkeiten die Referenzfarbe an die entsprechende `F_VN_ReferenceColorSimilarity` Funktion zu übergeben:

- Als trainiertes Farbmodell vom Typ `ITcVnColorModel`
- Als trainiertes Farbmodell vom Typ `ITcVnMlModel`
- Direkt mit Werten in einem `TcVnVector3_LREAL`

Für die Farbmodelle gibt es entsprechende Funktionen zum Trainieren und für alle drei Datentypen jeweils eine Funktion zur Ausführung:

- F_VN_TrainImageColor [▶ 1290] zum Trainieren eines Farbmodells
- F_VN_ReferenceColorSimilarity [▶ 1283] zur Segmentierung mittels Farbmodell

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.28 F_VN_TrainImageColorExp2_ITcVnMlModel

Create a new color model, describing the image color. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_TrainImageColorExp2_ITcVnMlModel : HRESULT
VAR_INPUT
    ipSrcImage           : ITcVnImage;
    ipColorModel         : Reference To ITcVnMlModel;
    nDifferentColors     : UDINT;
    eMethod              : ETcVnColorTrainingMethod;
    ipMask               : ITcVnImage;
    nSkipPixels          : UDINT;
    eClusteringAlgorithm : ETcVnClusteringAlgorithm;
    nMaxClusterRadius    : DINT;
    bSingleSplitSteps    : BOOL;
    hrPrev               : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (3 channels (RGB) of type USINT)
ipColorModel	Reference To ITcVnMlModel [▶ 435]	Returns the color model
nDifferentColors	UDINT	Maximum number of different colors to distinguish (if LBG is used as a clustering algorithm, the result might have less different colors, depending on nMaxClusterRadius)
eMethod	ETcVnColorTrainingMethod [▶ 198]	Color training method
ipMask	ITcVnImage [▶ 426]	Optional image mask (1 channel of type USINT, set to 0 if not required)
nSkipPixels	UDINT	Number of pixels to skip between each evaluated color sample (to achieve a better performance). 0 takes every pixel into account and tends to be more accurate.
eClusteringAlgorithm	ETcVnClusteringAlgorithm [▶ 180]	Clustering algorithm
nMaxClusterRadius	DINT	Only used for the LBG clustering algorithm. Maximum allowed radius (> 0) of a single cluster, i.e. clusters with a higher radius will be split into smaller ones, until a global number of nDifferentColors is reached.
bSingleSplitSteps	BOOL	Only used for the LBG clustering algorithm. If true, the global optimization is always run after a single cluster has been split. If false, several clusters are split within the same step before applying the global optimization. Applying the global optimization less often is faster, but can lead to less optimal results, especially having 2 nearby clusters that could be represented by 1)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_TrainImageColorExp2_ITcVnMlModel` ist die erweiterte Expert-Variante von F_VN_TrainImageColorExp_ITcVnMlModel [▶ 1295]. Sie enthält weitere Parameter. Das Farbmodel kann mit den entsprechenden Funktionsbausteinen auf der Festplatte gespeichert und geladen werden.

Parameter

Referenzbild

Das Referenzbild `ipSrcImage` muss 3 Kanäle im RGB-Format und den Element-Typ `USINT` (8 Bit) haben.

Farbmodell

Der Parameter `ipColorModel` gibt das trainierte Farbmodell zurück. Dieses ist ein Interface Pointer vom Typ ITcVnMlModel [▶ 435].

Anzahl zu unterscheidender Farben

Die Anzahl `nDifferentColors` definiert die Anzahl an zu trainierenden Farben. Achten Sie darauf, der Bildhintergrund z. B. auch eine eigene Farbe sein kann.

Clustering-Algorithmus

Bei der Auswahl des Clustering-Algorithmus stehen KMeans++ (feste Anzahl von Clustern) und LBG (dynamische Anzahl von Clustern) zur Verfügung.

Maximaler Cluster-Radius

Wird nur beim LBG-Clusteralgorithmus verwendet. Damit kann die Aufteilung der einzelnen Cluster beeinflusst werden.

Globale Optimierung

Wird nur beim LBG-Clusteralgorithmus verwendet. Damit kann zwischen kürzerer Ausführungszeit (`bSingleSplitSteps = FALSE`) und verbesserten Ergebnissen (`bSingleSplitSteps = TRUE`) gewählt werden.

Eine seltenere Anwendung der globalen Optimierung ist schneller, kann aber zu weniger optimalen Ergebnissen führen.

Anwendung

```
VAR
    ipColorModel    :   ITcVnColorModel;
END_VAR

hr := F_VN_TrainImageColorExp2_ITcVnMlModel(
            ipSrcImage          := ipImageRef,
            ipColorModel        := ipColorModel,
            nDifferentColors    := 5, // 4 colors + 1 background = 5
            eMethod             := TCVN_CTM_LAB,
            ipMask              := 0,
            nSkipPixels         := 3,
            eClusteringAlgorithm:= TCVN_CA_KMEANSPP,
            nMaxClusterRadius   := 0,
            bSingleSplitSteps   := FALSE,
            hrPrev              := hr);
```

Verwandte Funktionen

Es gibt drei Möglichkeiten die Referenzfarbe an die entsprechende `F_VN_ReferenceColorSimilarity` Funktion zu übergeben:

- Als trainiertes Farbmodell vom Typ `ITcVnColorModel`
- Als trainiertes Farbmodell vom Typ `ITcVnMlModel`
- Direkt mit Werten in einem `TcVnVector3_LREAL`

Für die Farbmodelle gibt es entsprechende Funktionen zum Trainieren und für alle drei Datentypen jeweils eine Funktion zur Ausführung:

- F_VN_TrainImageColor [▶ 1290] zum Trainieren eines Farbmodells
- F_VN_ReferenceColorSimilarity [▶ 1283] zur Segmentierung mittels Farbmodell

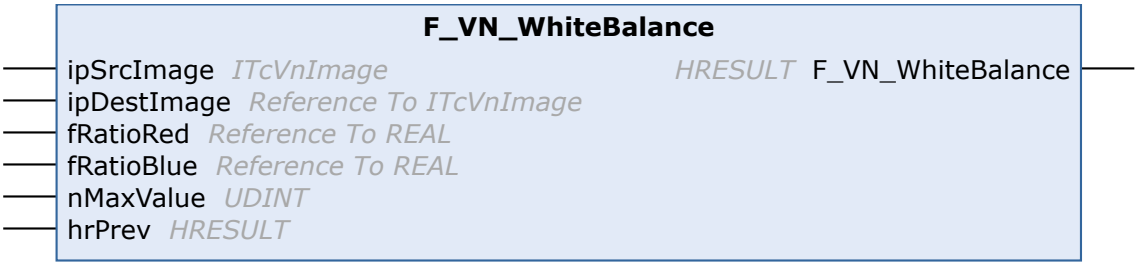
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.15.29 F_VN_WhiteBalance



Calculates the white balance ratios for a RGB image. Uses the green channel as a reference and computes the relative ratios for the red and blue channels. Requires a non-reflecting, not overexposed white colored object (e.g. a sheet of paper) in at least 100 pixels of the image to provide reasonable results. The resulting ratios can then be set on the camera (multiply with existing ratios), or applied in the PLC via MultiplyImageWithVector.

Syntax

Definition:

```
FUNCTION F_VN_WhiteBalance : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    fRatioRed   : Reference To REAL;
    fRatioBlue  : Reference To REAL;
    nMaxValue   : UDINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (3 channels (RGB), USINT or UINT)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image with white balance applied (optional, set to 0 if not required. An appropriate image will be created if required.)
fRatioRed	Reference To REAL	Returns the ratio for the red channel
fRatioBlue	Reference To REAL	Returns the ratio for the blue channel
nMaxValue	UDINT	Specify the maximum pixel value that would be achieved by overexposure (usually 255 for 8 bit or 4095 for 12 bit data)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16 Image Filtering

Diese Gruppe enthält Funktionen zur Filterung von Bildern.

Funktionen

Weichzeichnung

- F_VN_BilateralFilter(Exp) [▶ 1305]
- F_VN_BoxFilter(Exp) [▶ 1310]
- F_VN_GaussianFilter(Exp) [▶ 1324]
- F_VN_MedianFilter [▶ 1335]

Kantendetektion

- F_VN_LaplacianFilter(Exp) [▶ 1328]
- F_VN_ScharrFilter(Exp) [▶ 1342]
- F_VN_SobelFilter(Exp) [▶ 1350]

Benutzerdefinierte Filter

- F_VN_CustomFilter(Exp) [▶ 1316]
- F_VN_SeparableCustomFilter(Exp) [▶ 1346]

Flächen füllen

- F_VN_BrightBorderObjects [▶ 1312]
- F_VN_DarkBorderObjects [▶ 1321]
- F_VN_FillHoles [▶ 1322]

Morphologischer Operator

- F_VN_CreateStructuringElement [▶ 1314]
- F_VN_MorphologicalOperator [▶ 1338]

Lokale Extrema

- F_VN_LocalMaxima [▶ 1333]
- F_VN_LocalMinima [▶ 1334]
- F_VN_RemoveLocalMaxima [▶ 1339]
- F_VN_RemoveLocalMinima [▶ 1341]

Lokale Varianz

- F_VN_VarianceFilter(Exp) [▶ 1354]

Beispiele

Image Filtering [▶ 2892]

6.1.4.16.1 F_VN_BilateralFilter

Apply a Bilateral filter to smooth the image but preserve edges.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_BilateralFilter : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    nDiameter   : DINT;
    fSigmaColor : LREAL;
    fSigmaSpace : LREAL;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (Must be different from ipSrcImage! An appropriate destination image will be created if required.)
nDiameter	DINT	Diameter of the pixel neighborhood used for filtering (e.g. 5, 7, 9). If <= 0, it is automatically chosen dependend on fSigmaSpace.
fSigmaColor	LREAL	Sigma used for color space filtering (> 0). A larger value means that farther colors are mixed together.
fSigmaSpace	LREAL	Sigma used for coordinate space filtering (> 0). A larger value means that farther pixels can influence each other.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_BilateralFilter` wendet einen bilateralen Filter auf das Originalbild an. Der Filter glättet das Bild und reduziert Rauschen und erhält dabei die Kanten im Bild.

Parameter

Originalbild

Das Originalbild `ipSrcImage` muss vom Element-Typ `TCVN_ET_USINT` oder einem Fließkomma-Typ sein und 1 oder 3 Kanäle besitzen.

Ergebnisbild

Das Ergebnisbild `ipDestImage` muss ein anderer Interface Pointer als das Originalbild `ipSrcImage` sein!

Filter-Durchmesser

Der Filter-Durchmesser `nDiameter` bestimmt die Größe eines Filterbereichs. Es wird ein ungerader Wert erwartet. Wenn `nDiameter = 0` ist, dann wird der Filter-Durchmesser automatisch auf Basis von `fSigmaSpace` berechnet.

Farb-Bereich

Dieser Wert darf nicht `0` sein.

Raum-Bereich

Dieser Wert darf nicht `0` sein.

Expert-Parameter

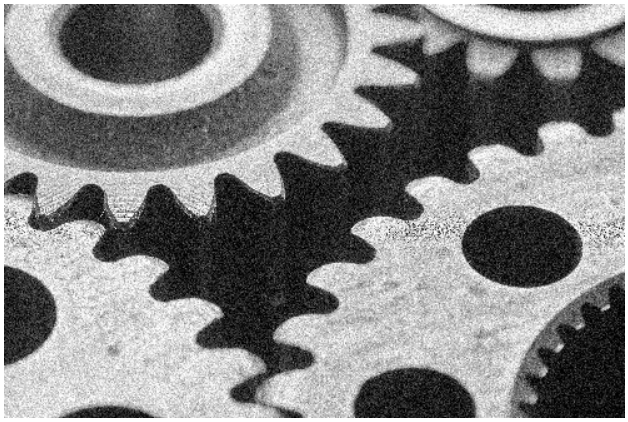
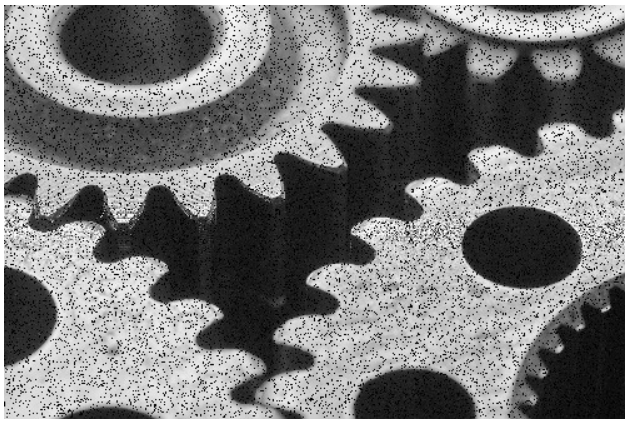
Weitere Parameter finden Sie in der Expert-Variante F_VN_BilateralFilterExp [▶ 1309].

Anwendung

Eine Anwendung dieser Funktion kann z. B. wie folgt aussehen:

```
hr := F_VN_BilateralFilter(
    ipSrcImage      :=  ipImage,
    ipDestImage     :=  ipImageRes,
    nDiameter       :=  nDiameter,
    fSigmaColor     :=  fSigmaColor,
    fSigmaSpace     :=  fSigmaSpace,
    hrPrev          :=  hr
);
```

Das unbearbeitete Originalbild (1. Zeile) weist im Detail bereits feine Strukturen auf den Oberflächen der Zahnräder auf. Um die Filterwirkungen zu verdeutlichen wurden zusätzliche Störungen durch ein Gauss- (2. Zeile) und ein Salz- und Pfeffer-Rauschen (3. Zeile) zu dem Originalbild hinzugefügt.

Originalbilder	Ergebnisbilder: Diameter = 7 Sigma = 100
	
	

Beispiele

- Weichzeichnungsfilter [▶ 2892]

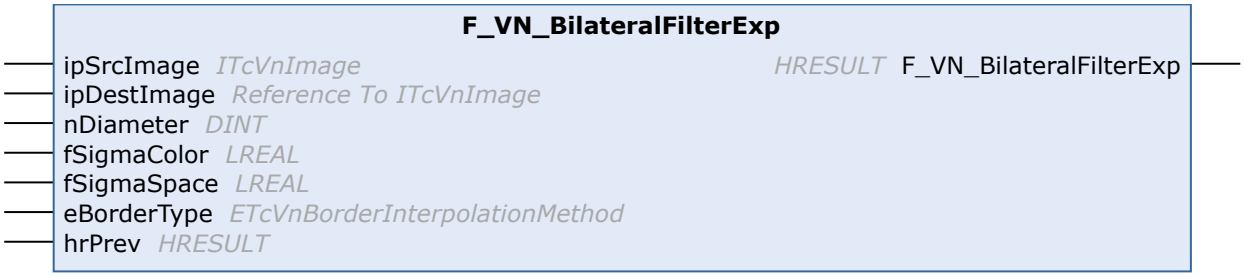
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.2 F_VN_BilateralFilterExp



Apply a Bilateral filter to smooth the image but preserve edges. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_BilateralFilterExp : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    nDiameter   : DINT;
    fSigmaColor : LREAL;
    fSigmaSpace : LREAL;
    eBorderType : ETcVnBorderInterpolationMethod;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (Must be different from ipSrcImage! An appropriate destination image will be created if required.)
nDiameter	DINT	Diameter of the pixel neighborhood used for filtering (e.g. 5, 7, 9). If <= 0, it is automatically chosen dependend on fSigmaSpace.
fSigmaColor	LREAL	Sigma used for color space filtering (> 0). A larger value means that farther colors are mixed together.
fSigmaSpace	LREAL	Sigma used for coordinate space filtering (> 0). A larger value means that farther pixels can influence each other.
eBorderType	ETcVnBorderInterpolationMethod [▶ 177]	Image border handling
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_BilateralFilterExp` ist die Expert-Variante von F_VN_BilateralFilter [▶ 1305]. Sie enthält zusätzliche Parameter.

Parameter

Originalbild

Das Originalbild `ipSrcImage` muss vom Element-Typ `TCVN_ET_USINT` oder einem Fließkomma-Typ sein und 1 oder 3 Kanäle besitzen.

Ergebnisbild

Das Ergebnisbild `ipDestImage` muss ein anderer Interface Pointer als das Originalbild `ipSrcImage` sein!

Filter-Durchmesser

Der Filter-Durchmesser `nDiameter` bestimmt die Größe eines Filterbereichs. Es wird ein ungerader Wert erwartet. Wenn `nDiameter = 0` ist, dann wird der Filter-Durchmesser automatisch auf Basis von `fSigmaSpace` berechnet.

Farb-Bereich

Dieser Wert darf nicht `0` sein.

Raum-Bereich

Dieser Wert darf nicht `0` sein.

Rand-Extrapolation

Die Art der Rand-Extrapolation `eBorderType` definiert, wie nichtexistente Pixel über die Bildgrenze hinaus extrapoliert werden, um die Filterwerte an den Bildgrenzen auszurechnen. Für weitere Details siehe ETcVnBorderInterpolationMethod [▶ 177].

Anwendung

Eine Anwendung dieser Funktion kann z. B. wie folgt aussehen:

```
hr := F_VN_BilateralFilterExp(
    ipSrcImage      :=  ipImage,
    ipDestImage     :=  ipImageRes,
    nDiameter       :=  nDiameter,
    fSigmaColor     :=  fSigmaColor,
    fSigmaSpace     :=  fSigmaSpace,
    eBorderType     :=  eBorderType,
    hrPrev          :=  hr
);
```

Beispiele

- Weichzeichnungsfilter [▶ 2892]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.3 F_VN_BoxFilter

Apply a box filter to an image.

Syntax

Definition:

```
FUNCTION F_VN_BoxFilter : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipDestImage   : Reference To ITcVnImage;
    nFilterWidth  : UDINT;
    nFilterHeight : UDINT;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
nFilterWidth	UDINT	Filter width in pixels
nFilterHeight	UDINT	Filter height in pixels
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

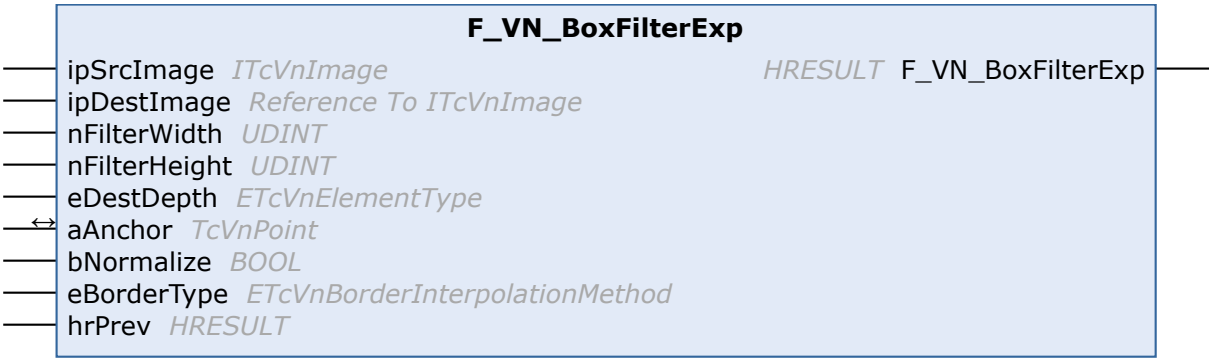
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.4 F_VN_BoxFilterExp



Apply a box filter to an image. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_BoxFilterExp : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipDestImage   : Reference To ITcVnImage;
```

```
    nFilterWidth  : UDINT;
    nFilterHeight : UDINT;
    eDestDepth    : ETcVnElementType;
END_VAR
VAR_IN_OUT
    aAnchor       : TcVnPoint;
END_VAR
VAR_INPUT
    bNormalize    : BOOL;
    eBorderType   : ETcVnBorderInterpolationMethod;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
nFilterWidth	UDINT	Filter width in pixels
nFilterHeight	UDINT	Filter height in pixels
eDestDepth	ETcVnElementType [▶ 210]	Destination image depth
bNormalize	BOOL	If true, the kernel is normalized by nFilterWidth * nFilterHeight
eBorderType	ETcVnBorderInterpolation Method [▶ 177]	Image border handling
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aAnchor	TcVnPoint [▶ 171]	Anchor point of the kernel ([-1, -1] for center)

Return value

HRESULT [▶ 150]

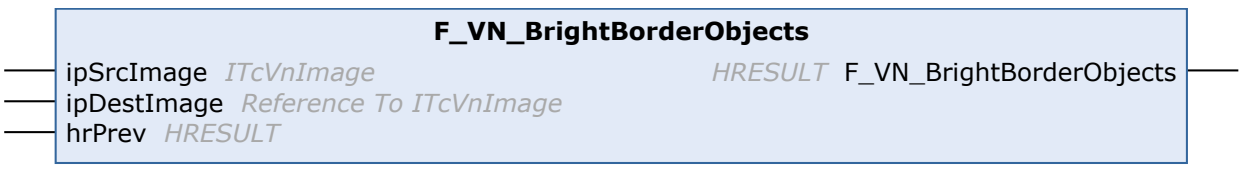
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.5 F_VN_BrightBorderObjects



Find bright objects connected to the image border within a gray-scale single-channel image.

Syntax

Definition:

```
FUNCTION F_VN_BrightBorderObjects : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Diese Funktion überträgt alle hellen Randobjekte im Eingangsbild auf das Ergebnisbild.

Algorithmus

Alle Objekte, die im Inneren des Bildes heller sind als die umgebenden Pixel, werden entfernt.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` darf nur einen Kanal haben und muss einen der folgenden Element-Typen haben: `USINT`, `UINT`, `INT`, `REAL` oder `LREAL`.

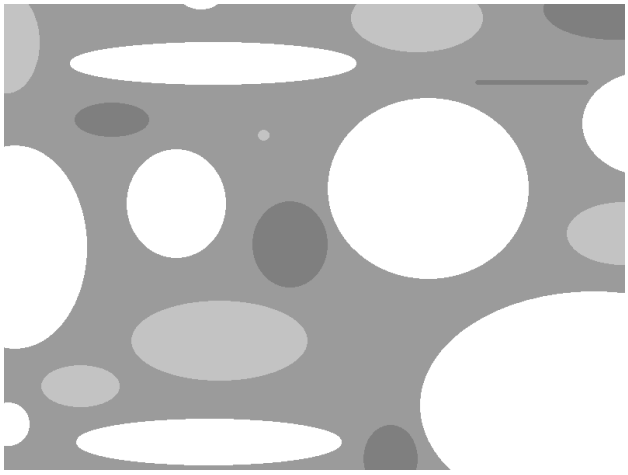
Ergebnisbild

Das Ergebnisbild `ipDestImage` gibt die hellen Randobjekte zurück.

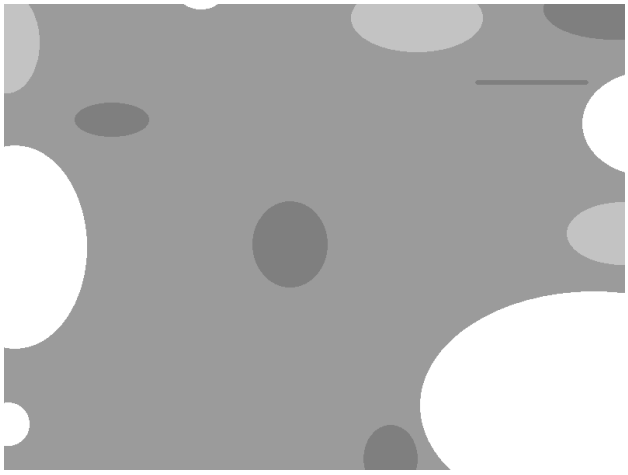
Anwendung

```
hr := F_VN_BrightBorderObjects(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    hrPrev          :=  hr
);
```

Eingangsbild



Ergebnisbild



Verwandte Funktionen

- F_VN_DarkBorderObjects [▶ 1321]
- F_VN_FillHoles [▶ 1322]

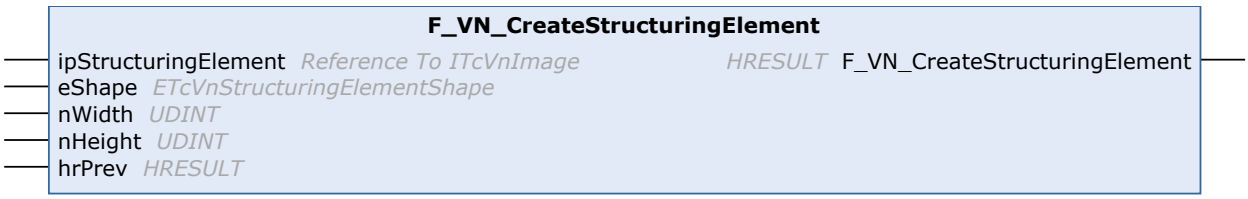
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.6 F_VN_CreateStructuringElement



Creates a structuring element for the usage with morphological operators and allocate an appropriate data buffer. The initial reference count is set to one if a new image interface is created and kept, otherwise.

Syntax

Definition:

```
FUNCTION F_VN_CreateStructuringElement : HRESULT
VAR_INPUT
    ipStructuringElement : Reference To ITcVnImage;
    eShape               : ETcVnStructuringElementShape;
    nWidth               : UDINT;
    nHeight              : UDINT;
    hrPrev               : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipStructuringElement	Reference To ITcVnImage [▶ 426]	Returns the created structuring element (Non-zero interface pointers are reused.)
eShape	ETcVnStructuringElementShape [▶ 229]	Shape of the structuring element (rectangle, cross, or ellipse)
nWidth	UDINT	Width
nHeight	UDINT	Height
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion erstellt ein Struktur-Element, welches für morphologische Operationen mit F_VN_MorphologicalOperator [▶ 1338] genutzt werden kann.

Parameter

Struktur-Element

Das erstellte Struktur-Element `ipStructuringElement` wird als ITcVnImage [▶ 426] zurückgegeben, in dem das Struktur-Element durch Pixel mit dem Wert `1` dargestellt wird.

Form

Die Form `eShape` des Struktur-Elements wird durch das Enum ETcVnStructuringElementShape [▶ 229] festgelegt. Folgende Formen sind verfügbar:

TCVN_SES_RECTANGLE

TCVN_SES_CROSS

TCVN_SES_ELLIPSE

Da die Pixel-Werte des Struktur-Elements 1 sind, wurden sie hier für eine bessere Anzeige hochskaliert.

Größe

Die Größe des Struktur-Elements wird durch die Breite `nWidth` und Höhe `nHeight` des zu erstellenden Bildes `ipStructuringElement` definiert.

Anwendung

Die Erstellung eines quadratischen Struktur-Elements der Größe `9` sieht z. B. wie folgt aus:

```
hr := F_VN_CreateStructuringElement(
    ipStructuringElement    :=  ipElement,
    eShape                  :=  TCVN_SES_RECTANGLE,
    nWidth                  :=  9,
    nHeight                 :=  9,
    hrPrev                  :=  hr
);
```

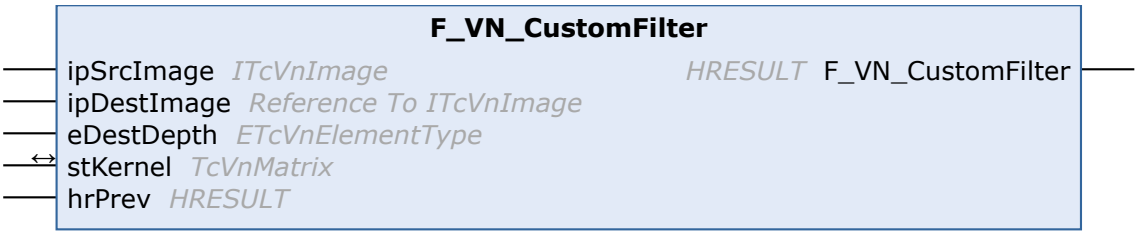
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.7 F_VN_CustomFilter



Apply a custom filter to the image.

Syntax

Definition:

```
FUNCTION F_VN_CustomFilter : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    eDestDepth  : ETcVnElementType;
END_VAR
VAR_IN_OUT
    stKernel    : TcVnMatrix;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 210]	Destination image depth
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stKernel	TcVnMatrix [▶ 245]	Custom filter kernel with values of type REAL or LREAL

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CustomFilter` wendet einen benutzerdefinierten Filter auf das Eingangsbild an. Der Filter-Kernel wird durch eine Matrix definiert. Diese Matrix wird mit dem Bild gefaltet.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` darf jedes verfügbare Format haben.

Ergebnisbild

Das Ergebnisbild `ipDestImage` hat das gleiche Format wie das Eingangsbild `ipSrcImage`.

Ergebnistiefe

Die Ergebnistiefe `eDestDepth` definiert, welchen Element-Typ das Ergebnisbild `ipDestImage` hat. Ein größerer Element-Typ kann mehr Informationen darstellen.

Filter-Kernel

Der benutzerdefinierte Filter-Kernel `stKernel` wird als Matrix übergeben. Erlaubte Element-Typen der Matrix sind `TCVN_ET_REAL` und `TCVN_ET_LREAL`. Eine entsprechende Matrix vom Typ TcVnMatrix [▶ 245] kann mit Hilfe der Funktion F_VN_InitMatrixStruct [▶ 1570] erstellt werden.

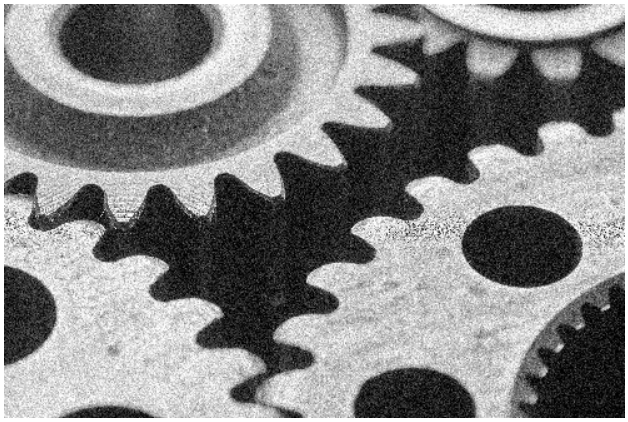
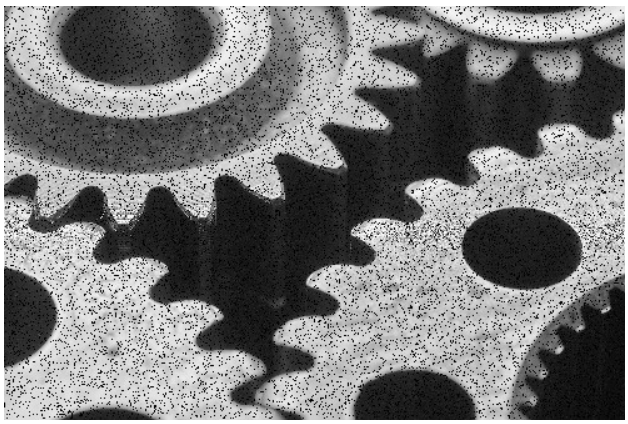
Expert-Parameter

Die Expert-Variante F_VN_CustomFilterExp [▶ 1319] enthält zusätzliche Parameter.

Anwendung

```
hr := F_VN_InitMatrixStruct(
    pSrcBuffer      := ADR(aMatrixArray7x7),
    stDestMatrix    := stKernelMatrix,
    nRows           := 7,
    nCols           := 7,
    eElementType    := TCVN_ET_REAL,
    hrPrev          := hr
);

hr := F_VN_CustomFilter(
    ipSrcImage  :=  ipImageIn,
    ipDestImage :=  ipImageRes,
    eDestDepth  :=  TCVN_ET_USINT,
    stKernel    :=  stKernelMatrix,
    hrPrev      :=  hr
);
```

- Das unbearbeitete Originalbild (1. Zeile) weist im Detail bereits feine Strukturen auf den Oberflächen der Zahnräder auf. Um die Filterwirkungen zu verdeutlichen wurden zusätzliche Störungen durch ein Gauss- (2. Zeile) und ein Salz- und Pfeffer-Rauschen (3. Zeile) zu dem Originalbild hinzugefügt. Als Beispiel wurde ein Mittelwertfilter umgesetzt, eine nähere Beschreibung befindet sich dazu im Beispiel der Weichzeichnungsfilter [▶ 2892]:

Originalbilder	Ergebnisbilder: Mean-Filter
	
	

Beispiele

- Weichzeichnungsfilter [▶ 2892]
- Benutzerdefinierte Filter

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.8 F_VN_CustomFilterExp

Apply a custom filter to the image. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_CustomFilterExp : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    eDestDepth  : ETcVnElementType;
END_VAR
VAR_IN_OUT
    stKernel    : TcVnMatrix;
    aAnchor     : TcVnPoint;
END_VAR
VAR_INPUT
    fDelta      : LREAL;
    eBorderType : ETcVnBorderInterpolationMethod;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 210]	Destination image depth
fDelta	LREAL	Value added to each pixel after filtering
eBorderType	ETcVnBorderInterpolationMethod [▶ 177]	Image border handling
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stKernel	TcVnMatrix [▶ 245]	Custom filter kernel with values of type REAL or LREAL
aAnchor	TcVnPoint [▶ 171]	Anchor point of the kernel

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CustomFilterExp` ist die Expert-Variante von F_VN_CustomFilter [▶ 1316]. Sie enthält zusätzliche Parameter.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` darf jedes verfügbare Format haben.

Ergebnisbild

Das Ergebnisbild `ipDestImage` hat das gleiche Format wie das Eingangsbild `ipSrcImage`.

Ergebnistiefe

Die Ergebnistiefe `eDestDepth` definiert, welchen Element-Typ das Ergebnisbild `ipDestImage` hat. Ein größerer Element-Typ kann mehr Informationen darstellen.

Filter-Kernel

Der benutzerdefinierte Filter-Kernel `stKernel` wird als Matrix übergeben. Erlaubte Element-Typen der Matrix sind `TCVN_ET_REAL` und `TCVN_ET_LREAL`. Eine entsprechende Matrix vom Typ TcVnMatrix [▶ 245] kann mit Hilfe der Funktion F_VN_InitMatrixStruct [▶ 1570] erstellt werden.

Ankerpunkt

Der Ankerpunkt `aAnchor` definiert, welcher Punkt des Filter-Kernels über jedes Pixel des Eingangsbildes `ipSrcImage` bewegt wird.

Delta

Das konstante Delta `fDelta` wird auf das Ergebnis der Filter-Operation addiert.

Rand-Extrapolation

Die Art der Rand-Extrapolation `eBorderType` definiert, wie nichtexistente Pixel über die Bildgrenze hinaus extrapoliert werden, um die Filterwerte an den Bildgrenzen auszurechnen. Für weitere Details siehe ETcVnBorderInterpolationMethod [▶ 177].

Anwendung

```
hr := F_VN_CustomFilterExp(
    ipSrcImage  :=  ipImageIn,
    ipDestImage :=  ipImageRes,
    eDestDepth  :=  TCVN_ET_USINT,
    stKernel    :=  stKernelMatrix,
    aAnchor     :=  aAnchorPoint,
    fDelta      :=  0,
    eBorderType :=  TCVN_BIM_DEFAULT,
    hrPrev      :=  hr
);
```

Beispiele

- Benutzerdefinierte Filter

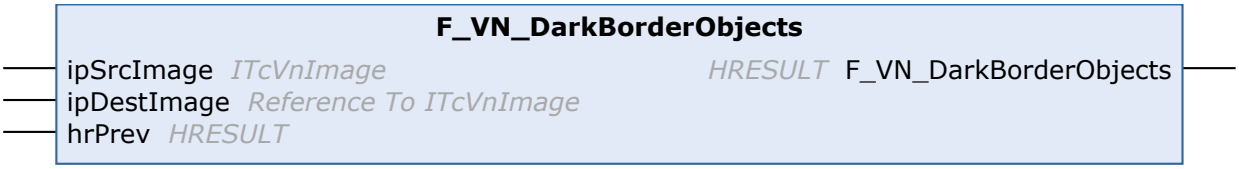
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.9 F_VN_DarkBorderObjects



Find dark objects connected to the image border within a gray-scale single-channel image. (equivalent to filling holes)

Syntax

Definition:

```
FUNCTION F_VN_DarkBorderObjects : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Diese Funktion überträgt alle dunklen Randobjekte im Eingangsbild auf das Ergebnisbild.

Algorithmus

Alle Objekte, die im Inneren des Bilds dunkler sind als die umgebenden Pixel, werden entfernt.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` darf nur einen Kanal haben und muss einen der folgenden Element-Typen haben: `USINT`, `UINT`, `INT`, `REAL` oder `LREAL`.

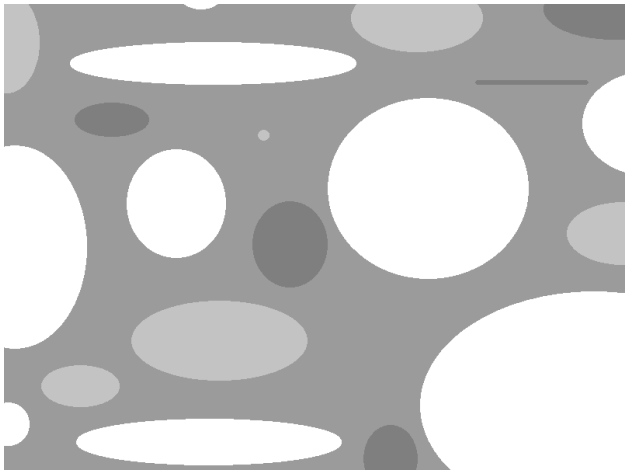
Ergebnisbild

Das Ergebnisbild `ipDestImage` gibt die dunklen Randobjekte zurück.

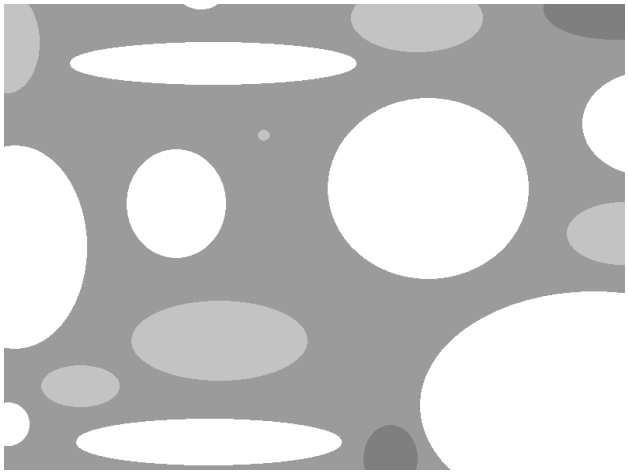
Anwendung

```
hr := F_VN_DarkBorderObjects(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    hrPrev          :=  hr
);
```

Eingangsbild



Ergebnisbild



Verwandte Funktionen

- F_VN_FillHoles [▶ 1322] als Alias
- F_VN_BrightBorderObjects [▶ 1312]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.10 F_VN_FillHoles

Fill holes within a gray-scale single-channel image. (equivalent to finding dark border objects)

Syntax

Definition:

```
FUNCTION F_VN_FillHoles : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Diese Funktion füllt Löcher in dem Eingangsbild.

Algorithmus

Alle Objekte, die im Inneren des Bilds dunkler sind als die umgebenden Pixel, werden entfernt.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` darf nur einen Kanal haben und muss einen der folgenden Element-Typen haben: `USINT`, `UINT`, `INT`, `REAL` oder `LREAL`.

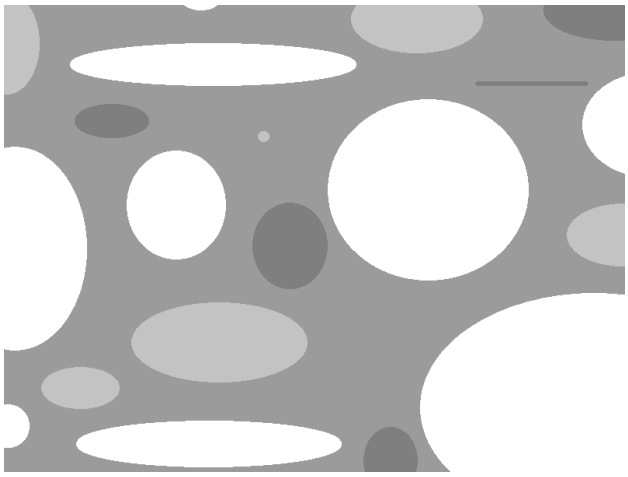
Ergebnisbild

Das Ergebnisbild `ipDestImage` gibt das Bild mit gefüllten Löchern zurück.

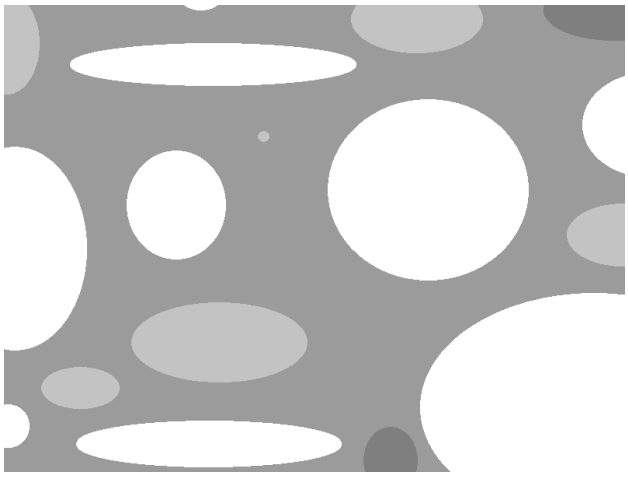
Anwendung

```
hr := F_VN_FillHoles(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    hrPrev          :=  hr
);
```

Eingangsbild



Ergebnisbild



Verwandte Funktionen

- F_VN_DarkBorderObjects [▶ 1321] als Alias
- F_VN_BrightBorderObjects [▶ 1312]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.11 F_VN_GaussianFilter

Apply a Gaussian filter to smooth the image.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_GaussianFilter : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipDestImage   : Reference To ITcVnImage;
    nFilterWidth  : UDINT;
    nFilterHeight : UDINT;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
nFilterWidth	UDINT	Filter width in pixels (1, 3, 5, 7, ...)
nFilterHeight	UDINT	Filter height in pixels (1, 3, 5, 7, ...)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_GaussianFilter` wendet einen Gauß-Filter auf das Eingangsbild an.

Algorithmus

Der Gauß-Filter wird in der Bildverarbeitung häufig zur Rauschunterdrückung bzw. Glättung eines Bildes eingesetzt. Dabei wird jedes Pixel durch den gewichteten Mittelwert seiner Umgebung ersetzt, wodurch kleinere Details entfernt werden, größere jedoch erhalten bleiben. Je größer der Filter gewählt wird, desto stärker wird das Bild geglättet (und wirkt dementsprechend unschärfer).

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` darf jedes verfügbare Format haben.

Ergebnisbild

Das Ergebnisbild `ipDestImage` erhält das gleiche Format wie das Eingangsbild `ipSrcImage`.

Filter-Größe

Die Größe des Gauß-Filters wird in X-Richtung durch `nFilterWidth` und in Y-Richtung durch `nFilterHeight` beschrieben. Die Größenangaben müssen ungerade sein, da es immer ein mittleres Pixel geben muss.

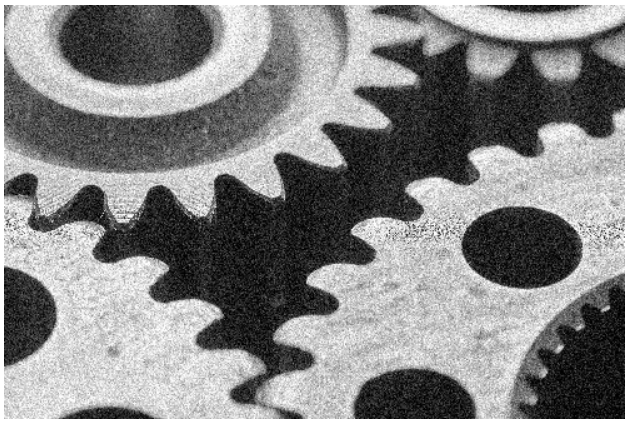
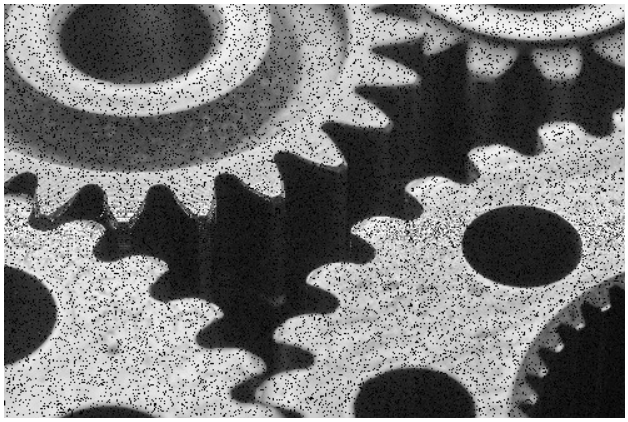
Expert-Parameter

Die Expert-Variante F_VN_GaussianFilterExp [▶ 1327] enthält zusätzliche Parameter.

Anwendung

```
hr := F_VN_GaussianFilter(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    nFilterWidth    :=  7,
    nFilterHeight   :=  7,
    hrPrev          :=  hr
);
```

Das unbearbeitete Originalbild (1. Zeile) weist im Detail bereits feine Strukturen auf den Oberflächen der Zahnräder auf. Um die Filterwirkungen zu verdeutlichen wurden zusätzliche Störungen durch ein Gauss- (2. Zeile) und ein Salz- und Pfeffer-Rauschen (3. Zeile) zu dem Originalbild hinzugefügt.

Originalbilder	Ergebnisbilder
	
	

Beispiele

- Weichzeichnungsfilter [▶ 2892]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.12 F_VN_GaussianFilterExp

Apply a Gaussian filter to smooth the image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_GaussianFilterExp : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipDestImage   : Reference To ITcVnImage;
    nFilterWidth  : UDINT;
    nFilterHeight : UDINT;
    fSigmaX       : LREAL;
    fSigmaY       : LREAL;
    eBorderType   : ETcVnBorderInterpolationMethod;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
nFilterWidth	UDINT	Filter width in pixels (1, 3, 5, 7, ...)
nFilterHeight	UDINT	Filter height in pixels (1, 3, 5, 7, ...)
fSigmaX	LREAL	Gaussian kernel standard deviation in X direction (>= 0, automatically chosen if 0)
fSigmaY	LREAL	Gaussian kernel standard deviation in Y direction (>= 0, automatically chosen if 0)
eBorderType	ETcVnBorderInterpolationMethod [▶ 177]	Image border handling
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_GaussianFilterExp` ist die Expert-Variante von F_VN_GaussianFilter [▶ 1324]. Sie enthält zusätzliche Parameter.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` darf jedes verfügbare Format haben.

Ergebnisbild

Das Ergebnisbild `ipDestImage` erhält das gleiche Format wie das Eingangsbild `ipSrcImage`.

Filter-Größe

Die Größe des Gauß-Filters wird in X-Richtung durch `nFilterWidth` und in Y-Richtung durch `nFilterHeight` beschrieben. Die Größenangaben müssen ungerade sein, da es immer ein mittleres Pixel geben muss.

Standardabweichung

Die Standardabweichung in X-Richtung fSigmaX und in Y-Richtung fSigmaY definiert zusätzlich zur Filtergröße die Form der Gauß-Kurve, die im Filter verwendet wird.

Rand-Extrapolation

Die Art der Rand-Extrapolation `eBorderType` definiert, wie nichtexistente Pixel über die Bildgrenze hinaus extrapoliert werden, um die Filterwerte an den Bildgrenzen auszurechnen. Für weitere Details siehe ETcVnBorderInterpolationMethod [▶ 177].

Anwendung

```
hr := F_VN_GaussianFilterExp(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    nFilterWidth    :=  3,
    nFilterHeight   :=  3,
    fSigmaX         :=  5,
    fSigmaY         :=  5,
    eBorderType     :=  TCVN_BIM_DEFAULT,
    hrPrev          :=  hr
);
```

Beispiele

- Weichzeichnungsfilter [▶ 2892]

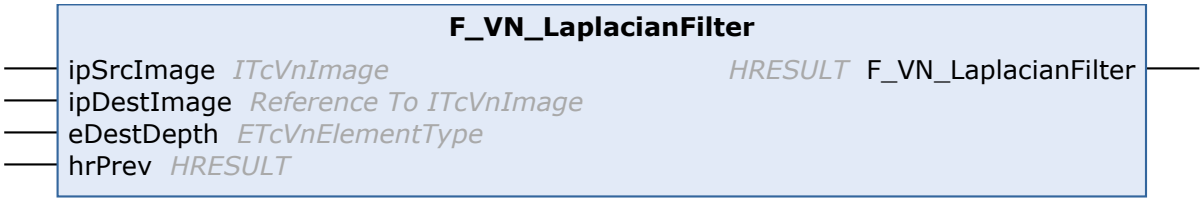
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.13 F_VN_LaplacianFilter



Apply a Laplacian filter to an image.

Syntax

Definition:

```
FUNCTION F_VN_LaplacianFilter : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    eDestDepth  : ETcVnElementType;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 210]	Destination image depth
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_LaplacianFilter` wendet einen Laplace-Filter auf das Eingangsbild an.

Algorithmus

Der Laplace-Filter kann verwendet werden um Kanten zu detektieren, indem die 2. Ableitung der Übergänge benachbarter Pixelintensitäten approximiert wird. Da die Ableitung auch negative Werte hat, empfiehlt es sich einen vorzeichenbehafteten Elementtyp für das Ergebnisbild zu verwenden (z.B. `TCVN_ET_INT`).

Für nKernelSize = 3, 5, ... wird der Laplace-Filter durch Addition der 2. Ableitung in x- und y-Richtung über die angegebene Fenstergröße berechnet. Für nKernelSize = 1 wird der folgende spezielle 3 x 3 Kernel verwendet:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Parameter

Eingangsbild

Das Eingangsbild ipSrcImage darf, außer den grundsätzlich nicht unterstützen Datentypen SINT und DINT, jedes verfügbare Format haben.

Ergebnisbild

Das Ergebnisbild ipDestImage hat die gleiche Größe wie das Eingangsbild ipSrcImage, das Pixelformat wird allerdings durch die Ergebnistiefe `eDestDepth` bestimmt.

Ergebnistiefe

Die Ergebnistiefe `eDestDepth` definiert, welchen Element-Typ das Ergebnisbild `ipDestImage` hat. Ein größerer Element-Typ kann mehr Informationen darstellen.

HINWEIS

Kombination von Eingangsbild und Ergebnistiefe

Als Ergebnistiefe kann immer nur der gleiche oder ein größerer Datentyp wie der des Eingangsbildes verwendet werden, da sonst Daten verloren gehen würden. Z. B. kann ein UINT nicht in ein USINT gewandelt werden. Weiterhin bestehen noch zwei Einschränkungen, so dass ein UINT nicht in ein INT und ein REAL nicht in ein LREAL gewandelt werden können.

Expert-Parameter

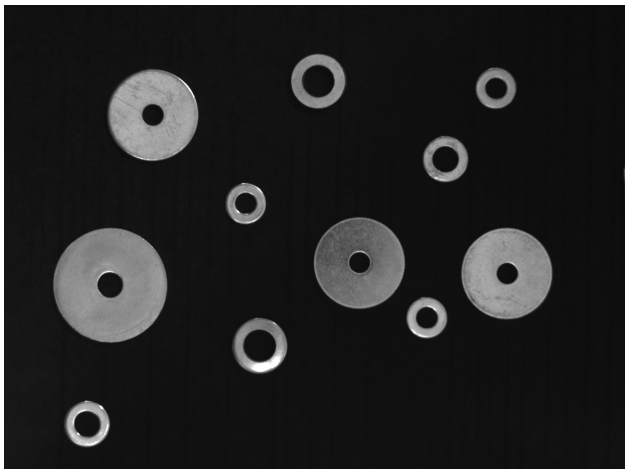
Die Expert-Variante F_VN_LaplacianFilterExp [▶ 1331] enthält zusätzliche Parameter.

Anwendung

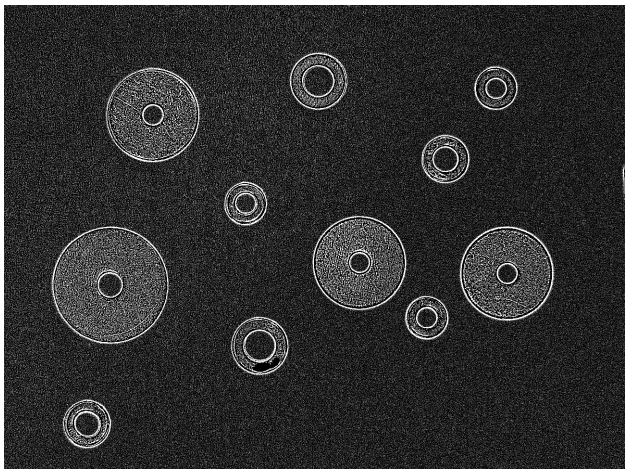
Das Anwenden eines Laplace-Filters mit einer Ergebnistiefe von 8 Bit sieht z. B. so aus:

```
hr := F_VN_LaplacianFilter(
    ipSrcImage  :=  ipImageIn,
    ipDestImage :=  ipImageRes,
    eDestDepth  :=  TCVN_ET_USINT,
    hrPrev      :=  hr
);
```

Originalbild



Ergebnisbild



Beispiele

- Kantendetektionsfilter [▶ 2894]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.14 F_VN_LaplacianFilterExp

Apply a Laplacian filter to an image. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_LaplacianFilterExp : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    eDestDepth  : ETcVnElementType;
    nKernelSize : UDINT;
    fScale      : LREAL;
    fDelta      : LREAL;
    eBorderType : ETcVnBorderInterpolationMethod;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 210]	Destination image depth
nKernelSize	UDINT	Aperture size used to compute the second-derivative filters (1, 3, 5, ..., 31)
fScale	LREAL	Scale factor for the computed derivative values
fDelta	LREAL	Delta value that is added to the results before storing them
eBorderType	ETcVnBorderInterpolationMethod [▶ 177]	Image border handling
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_LaplacianFilterExp` ist die Expert-Variante von F_VN_LaplacianFilter [▶ 1328]. Sie enthält zusätzliche Parameter.

Parameter

Eingangsbild

Das Eingangsbild ipSrcImage darf, außer den grundsätzlich nicht unterstützen Datentypen SINT und DINT, jedes verfügbare Format haben.

Ergebnisbild

Das Ergebnisbild ipDestImage hat die gleiche Größe wie das Eingangsbild ipSrcImage, das Pixelformat wird allerdings durch die Ergebnistiefe `eDestDepth` bestimmt.

Ergebnistiefe

Die Ergebnistiefe `eDestDepth` definiert, welchen Element-Typ das Ergebnisbild `ipDestImage` hat. Ein größerer Element-Typ kann mehr Informationen darstellen.

HINWEIS

Kombination von Eingangsbild und Ergebnistiefe

Als Ergebnistiefe kann immer nur der gleiche oder ein größerer Datentyp wie der des Eingangsbildes verwendet werden, da sonst Daten verloren gehen würden. Z. B. kann ein UINT nicht in ein USINT gewandelt werden. Weiterhin bestehen noch zwei Einschränkungen, so dass ein UINT nicht in ein INT und ein REAL nicht in ein LREAL gewandelt werden können.

Kernel-Größe

Der Parameter `nKernelSize` definiert die Größe des Filter-Kernels zur Approximation der 2. Ableitung. Es sind nur ungerade Werte zwischen `1` und `31` erlaubt.

Skalierung

Das Ergebnis der Laplace-Operation wird mit dem Skalierungsfaktor `fScale` multipliziert.

Delta

Das konstante Delta `fDelta` wird auf das Ergebnis der Scharr-Operation nach Skalierung mit `fScale` dazuaddiert.

Rand-Extrapolation

Die Art der Rand-Extrapolation `eBorderType` definiert, wie nichtexistente Pixel über die Bildgrenze hinaus extrapoliert werden, um die Filterwerte an den Bildgrenzen auszurechnen. Für weitere Details siehe ETcVnBorderInterpolationMethod [▶ 177].

Anwendung

Das Anwenden eines Laplace-Filters mit einer Kernel-Größe von `3×3` sieht z. B. so aus:

```
hr := F_VN_LaplacianFilterExp(
    ipSrcImage  :=  ipImageIn,
    ipDestImage :=  ipImageRes,
    eDestDepth  :=  TCVN_ET_USINT,
    nKernelSize :=  3,
    fScale      :=  1,
    fDelta      :=  0,
    eBorderType :=  TCVN_BIM_DEFAULT,
    hrPrev      :=  hr
);
```

Beispiele

- Kantendetektionsfilter [▶ 2894]

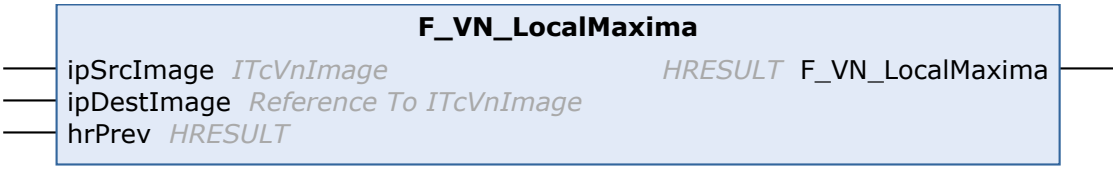
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.15 F_VN_LocalMaxima



Find local maxima in a gray-scale single-channel image. The found maxima are marked by a value of 1 in the destination image.

Syntax

Definition:

```
FUNCTION F_VN_LocalMaxima : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT or UINT, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (Must be different from ipSrcImage! An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_LocalMaxima` findet lokale Maxima im Eingangsbild.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Grauwertbild mit einer Bittiefe von 8 oder 16 (USINT oder UINT) sein.

Ergebnisbild

Das Ergebnisbild `ipDestImage` ist vom gleichen Typ wie das Eingangsbild `ipSrcImage`. Jedes lokale Maximum des Eingangsbilds ist hier durch einen Pixelwert von 1 markiert.

Anwendung

Das Finden von lokalen Maxima sieht so aus:

```
hr := F_VN_LocalMaxima(
    ipSrcImage  :=  ipImageIn,
    ipDestImage :=  ipImageRes,
    hrPrev      :=  hr
);
```

Beispiele

- Lokale Extrema entfernen

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.16 F_VN_LocalMinima

Find local minima in a gray-scale single-channel image. The found minima are marked by a value of 1 in the destination image.

Syntax

Definition:

```
FUNCTION F_VN_LocalMinima : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT or UINT, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (Must be different from ipSrcImage! An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_LocalMinima` findet lokale Minima im Eingangsbild.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Grauwertbild mit einer Bittiefe von 8 oder 16 (USINT oder UINT) sein.

Ergebnisbild

Das Ergebnisbild `ipDestImage` ist vom gleichen Typ wie das Eingangsbild `ipSrcImage`. Jedes lokale Minimum des Eingangsbilds ist hier durch einen Pixelwert von 1 markiert.

Anwendung

Das Finden von lokalen Minima sieht so aus:

```
hr := F_VN_LocalMinima(
    ipSrcImage  :=  ipImageIn,
    ipDestImage :=  ipImageRes,
    hrPrev      :=  hr
);
```

Beispiele

- Lokale Extrema entfernen

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.17 F_VN_MedianFilter

Apply a Median filter to an image.

Syntax

Definition:

```
FUNCTION F_VN_MedianFilter : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    nFilterSize : UDINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (for nFilterSize 3 or 5: USINT, UINT, REAL. For bigger filters, only USINT is supported.
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
nFilterSize	UDINT	Size (width and height) of the filter (3, 5, 7, ...)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

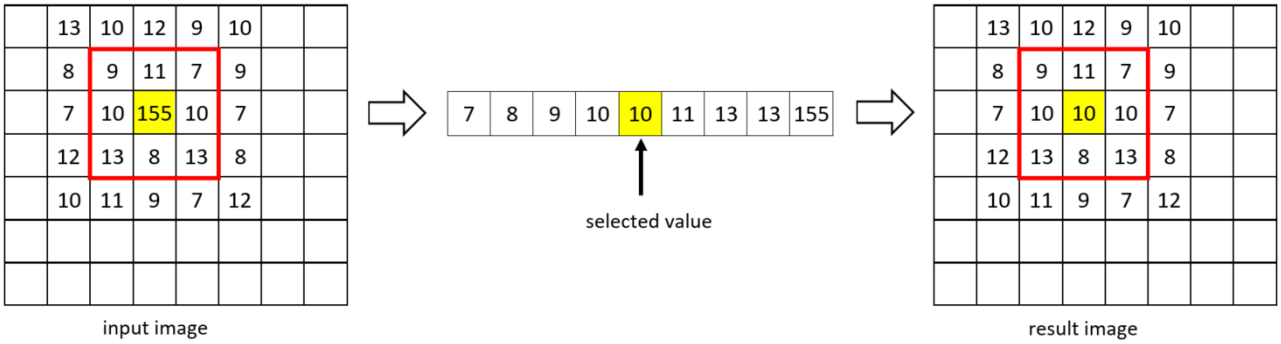
Weiterführende Informationen

Die Funktion `F_VN_MedianFilter` wendet einen Median-Filter auf das Eingangsbild an.

Algorithmus

Der Median-Filter ist ein Rangordnungsfilter, der häufig zur Rauschunterdrückung eingesetzt wird. Dabei werden alle von der Filtermatrix erfassten Pixelwerte der Größe nach sortiert und anschließend der aktuelle Wert durch den Wert an mittlerer Position der sortierten Liste ersetzt. Der Vorteil gegenüber einem Filter mit gewichtetem Mittelwert (z. B. einem Gauß-Filter) besteht darin, dass starke Ausreißer einzelner Pixel (wie z. B. die 155 im nachfolgenden Beispielbild) komplett ignoriert werden und somit das Ergebnis nicht negativ beeinflussen.

Die Größe der Filtermatrix wird über `nFilterSize` angeben, wobei nur ungerade Ganzzahlen `>=` 3 akzeptiert werden.



Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` kann beliebig viele Kanäle haben. Bei einer Filter-Größe `nFilterSize` von `3` oder `5` sind als Element-Typen `USINT`; `UINT` und `REAL` erlaubt. Für größere Filter wird nur der Element-Typ `USINT` unterstützt.

Ergebnisbild

Das Ergebnisbild `ipDestImage` gibt das gefilterte Bild zurück.

Filter-Größe

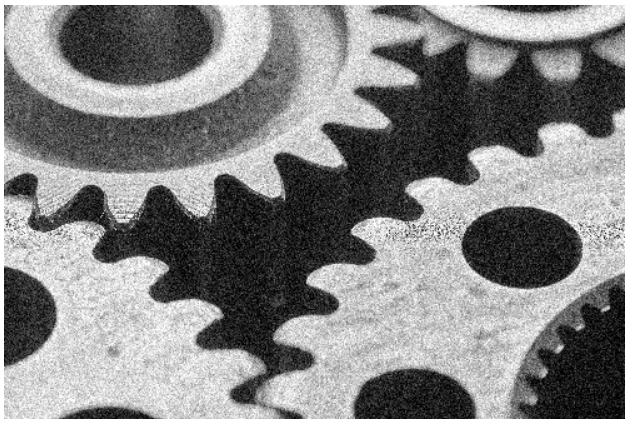
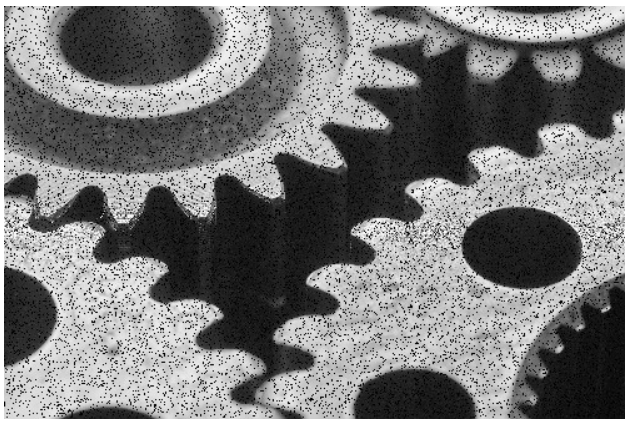
Die Filter-Größe `nFilterSize` legt die Höhe und Breite des Filterbereichs fest. Sie muss ungerade und `>=` `3` sein.

Anwendung

Die Anwendung eines Median-Filters mit der Größe `7×7` sieht z. B. so aus:

```
hr := F_VN_MedianFilter(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    nFilterSize     :=  7,
    hrPrev          :=  hr
);
```

Das unbearbeitete Originalbild (1. Zeile) weist im Detail bereits feine Strukturen auf den Oberflächen der Zahnräder auf. Um die Filterwirkungen zu verdeutlichen wurden zusätzliche Störungen durch ein Gauss- (2. Zeile) und ein Salz- und Pfeffer-Rauschen (3. Zeile) zu dem Originalbild hinzugefügt.

Originalbilder	Ergebnisbilder
	
	

Beispiele

- Weichzeichnungsfilter [▶ 2892]

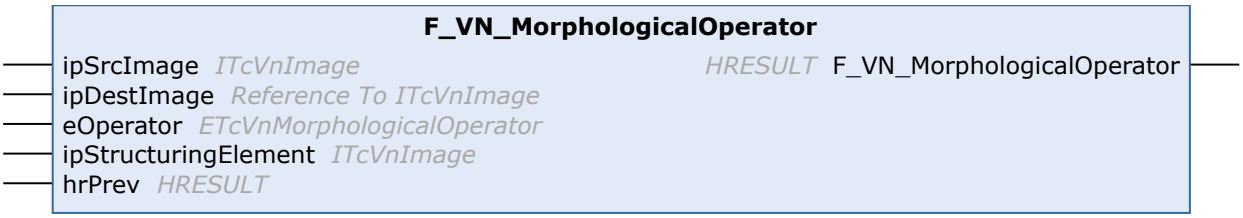
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.18 F_VN_MorphologicalOperator



Apply a morphological operator.

Syntax

Definition:

```
FUNCTION F_VN_MorphologicalOperator : HRESULT
VAR_INPUT
    ipSrcImage           : ITcVnImage;
    ipDestImage          : Reference To ITcVnImage;
    eOperator            : ETcVnMorphologicalOperator;
    ipStructuringElement : ITcVnImage;
    hrPrev               : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (only 1 channel supported for reconstruction operators)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
eOperator	ETcVnMorphologicalOperator [▶ 220]	Operator type
ipStructuringElement	ITcVnImage [▶ 426]	Structuring element to be applied (Typically created via F_VN_CreateStructuringElement.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_MorphologicalOperator` wendet einen morphologischen Operator auf das Eingangsbild an.

Parameter

Originalbild

Die Reconstruction-Operatoren können nur auf ein 1-Kanal-Bild angewendet werden, die anderen Operatoren stellen keine besonderen Anforderungen an das Originalbild `ipSrcImage`.

Ergebnisbild

Im Ergebnisbild `ipDestImage` wird das veränderte Bild zurückgegeben.

Morphologischer Operator

Der Parameter `eOperator` definiert, welcher morphologische Operator auf das Originalbild `ipSrcImage` angewandt wird. Zur Verfügung stehen alle Operatoren im Enum ETcVnMorphologicalOperator [▶ 220].

Struktur-Element

Das Struktur-Element `ipStructuringElement` ist das Element, das das Filtergebiet des morphologischen Operators vorgibt. Es kann mit Hilfe der Funktion F_VN_CreateStructuringElement [▶ 1314] erstellt werden.

Anwendung

Die Anwendung eines Opening-Operators mit einem rechteckigen Filtergebiet der Größe `3×3` sieht z. B. so aus:

```
VAR
    ipElement : ITcVnImage;
END_VAR

hr := F_VN_CreateStructuringElement(ipElement, TCVN_SES_RECTANGLE, 3, 3, hr);

hr := F_VN_MorphologicalOperator(
    ipSrcImage  :=  ipImageIn,
    ipDestImage :=  ipImageRes,
    eOperator   :=  TCVN_MO_OPENING,
    ipStructuringElement  :=  ipElement,
    hrPrev      :=  hr
);
```

Beispiele

- Morphologische Operatoren [▶ 2896]

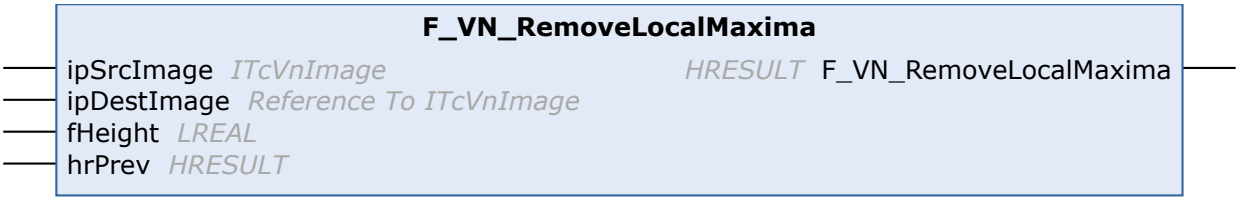
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.19 F_VN_RemoveLocalMaxima



Remove local maxima up to a given height from a gray-scale single-channel image.

Syntax

Definition:

```
FUNCTION F_VN_RemoveLocalMaxima : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    fHeight     : LREAL;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
fHeight	LREAL	Maximum height of the maxima to be removed (must be greater than 0)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_RemoveLocalMaxima` entfernt lokale Maxima aus dem Eingangsbild.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` darf nur einen Kanal haben und muss einen der folgenden Element-Typen haben: `USINT`, `UINT`, `INT`, `REAL` oder `LREAL`.

Ergebnisbild

Das Ergebnisbild `ipDestImage` gibt das Bild zurück, in dem lokale Maxima entfernt wurden.

Maximal-Höhe

Die Maximal-Höhe `fHeight` definiert, bis zu welcher Höhe Maxima im Eingangsbild entfernt werden. Die Höhe beschreibt dabei die Intensitätsdifferenz zwischen dem Maximum und dem Wert, auf den die betroffenen Pixel gesetzt werden. `fHeight` muss größer als `0` sein.

Anwendung

Das Entfernen von lokalen Maxima bis zu einer Höhe von `10` sieht z. B. so aus:

```
hr := F_VN_RemoveLocalMaxima(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    fHeight         :=  10,
    hrPrev          :=  hr
);
```

Beispiele

- Lokale Extrema entfernen

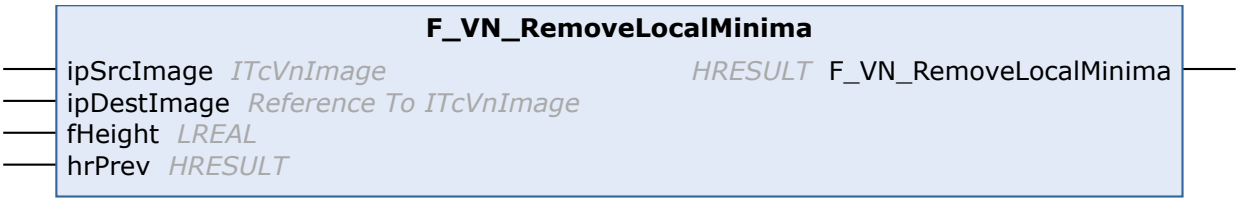
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.20 F_VN_RemoveLocalMinima



Remove local minima up to a given height from a gray-scale single-channel image.

Syntax

Definition:

```
FUNCTION F_VN_RemoveLocalMinima : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    fHeight     : LREAL;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
fHeight	LREAL	Maximum height of the minima to be removed (must be greater than 0)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_RemoveLocalMinima` entfernt lokale Minima aus dem Eingangsbild.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` darf nur einen Kanal haben und muss einen der folgenden Element-Typen haben: `USINT`, `UINT`, `INT`, `REAL` oder `LREAL`.

Ergebnisbild

Das Ergebnisbild `ipDestImage` gibt das Bild zurück, in dem die lokalen Minima entfernt wurden.

Maximal-Höhe

Die Maximal-Höhe `fHeight` definiert, bis zu welcher Höhe Minima im Eingangsbild entfernt werden. Die Höhe beschreibt dabei die Intensitätsdifferenz zwischen dem Minimum und dem Wert, auf den die betroffenen Pixel gesetzt werden. `fHeight` muss größer als `0` sein.

Anwendung

Das Entfernen von lokalen Minima bis zu einer Höhe von `10` sieht z.B. so aus:

```
hr := F_VN_RemoveLocalMinima(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    fHeight         :=  10,
    hrPrev          :=  hr
);
```

Beispiele

- Lokale Extrema entfernen

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.21 F_VN_ScharrFilter

Calculates the first order derivative in x or y direction using a Scharr filter.

Syntax

Definition:

```
FUNCTION F_VN_ScharrFilter : HRESULT
VAR_INPUT
    ipSrcImage       : ITcVnImage;
    ipDestImage      : Reference To ITcVnImage;
    eDestDepth       : ETcVnElementType;
    eFilterDirection : ETcVnFilterDirection;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 210]	Destination image depth
eFilterDirection	ETcVnFilterDirection [▶ 214]	Filter direction
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_ScharrFilter` wendet einen Scharr-Filter auf das Eingangsbild an. Der Scharr-Filter funktioniert analog zum Sobel-Filter [▶ 1350] und unterscheidet sich von diesem nur durch den Kernel.

Algorithmus

Der Scharr-Filter ist eng mit dem Sobel-Filter verwandt. Er liefert jedoch genauere Ergebnisse als ein 3 x 3 Sobel-Filter.

$$G_x = \begin{bmatrix} -3 & 0 & +3 \\ -10 & 0 & +10 \\ -3 & 0 & +3 \end{bmatrix} \qquad G_y = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ +3 & +10 & +3 \end{bmatrix}$$

Parameter

Originalbild

Das Originalbild `ipSrcImage` kann jedes verfügbare Format haben.

Ergebnisbild

Das Ergebnisbild `ipDestImage` liefert das Filter-Ergebnis zurück und besitzt das gleiche Format wie das Originalbild `ipSrcImage`.

Ergebnistiefe

Die Ergebnistiefe `eDestDepth` definiert, welchen Element-Typ das Ergebnisbild `ipDestImage` hat. Ein größerer Element-Typ kann mehr Informationen darstellen.

Filter-Richtung

Die Filter-Richtung `eFilterDirection` definiert, ob der Scharr-Filter in X-Richtung (`TCVN_FD_X`) oder in Y-Richtung (`TCVN_FD_Y`) auf das Bild angewandt wird.

Expert-Parameter

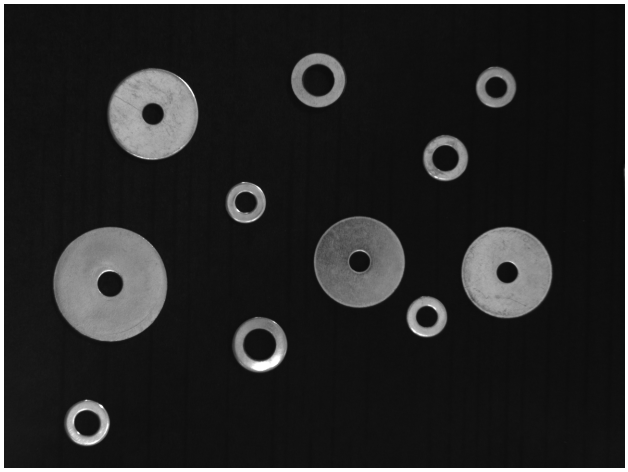
Weitere Parameter finden Sie in der Expert-Variante F_VN_ScharrFilterExp [▶ 1344].

Anwendung

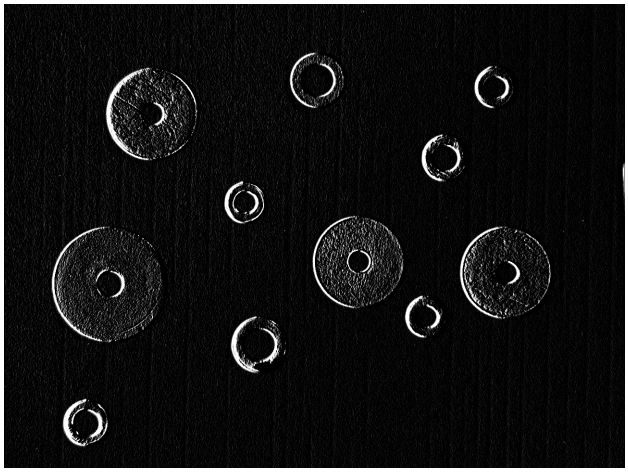
```
hr := F_VN_ScharrFilter(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    eDestDepth      :=  TCVN_ET_USINT,
```

```
    eFilterDirection:=  TCVN_FD_X,
    hrPrev          :=  hr,
);
```

Originalbild



Ergebnisbild



Beispiele

- Kantendetektionsfilter [▶ 2894]

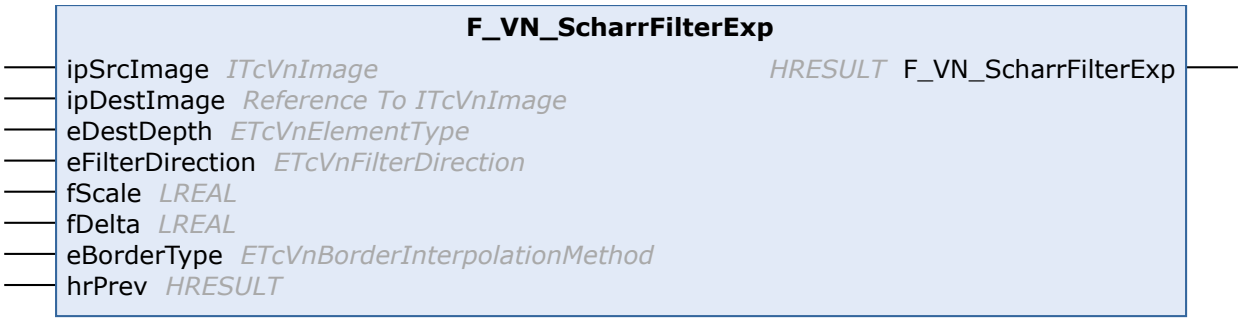
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.22 F_VN_ScharrFilterExp



Calculates the first order derivative in x or y direction using a Scharr filter. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_ScharrFilterExp : HRESULT
VAR_INPUT
    ipSrcImage       : ITcVnImage;
    ipDestImage      : Reference To ITcVnImage;
    eDestDepth       : ETcVnElementType;
    eFilterDirection : ETcVnFilterDirection;
    fScale           : LREAL;
    fDelta           : LREAL;
    eBorderType      : ETcVnBorderInterpolationMethod;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 210]	Destination image depth
eFilterDirection	ETcVnFilterDirection [▶ 214]	Filter direction
fScale	LREAL	Scale factor for the computed derivative values
fDelta	LREAL	Delta value that is added to the results prior to storing them in dest
eBorderType	ETcVnBorderInterpolationMethod [▶ 177]	Image border handling
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_ScharrFilterExp` ist die Expert-Variante von F_VN_ScharrFilter [▶ 1342]. Sie enthält zusätzliche Parameter.

Parameter

Originalbild

Das Originalbild `ipSrcImage` kann jedes verfügbare Format haben.

Ergebnisbild

Das Ergebnisbild `ipDestImage` liefert das Filter-Ergebnis zurück und besitzt das gleiche Format wie das Originalbild `ipSrcImage`.

Ergebnistiefe

Die Ergebnistiefe `eDestDepth` definiert, welchen Element-Typ das Ergebnisbild `ipDestImage` hat. Ein größerer Element-Typ kann mehr Informationen darstellen.

Filter-Richtung

Die Filter-Richtung `eFilterDirection` definiert, ob der Scharr-Filter in X-Richtung (`TCVN_FD_X`) oder in Y-Richtung (`TCVN_FD_Y`) auf das Bild angewandt wird.

Skalierung

Das Ergebnis der Scharr-Operation wird mit dem Skalierungsfaktor `fScale` multipliziert.

Delta

Das konstante Delta `fDelta` wird auf das Ergebnis der Scharr-Operation nach Skalierung mit `fScale` dazuaddiert.

Rand-Extrapolation

Die Art der Rand-Extrapolation `eBorderType` definiert, wie nichtexistente Pixel über die Bildgrenze hinaus extrapoliert werden, um die Filterwerte an den Bildgrenzen auszurechnen. Für weitere Details siehe ETcVnBorderInterpolationMethod [▶ 177].

Anwendung

```
hr := F_VN_ScharrFilterExp(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    eDestDepth      :=  TCVN_ET_USINT,
    eFilterDirection:=  TCVN_FD_X,
    fScale          :=  1,
    fDelta          :=  0,
    eBorderType     :=  TCVN_BIM_DEFAULT,
    hrPrev          :=  hr,
);
```

Beispiele

- Kantendetektionsfilter [▶ 2894]

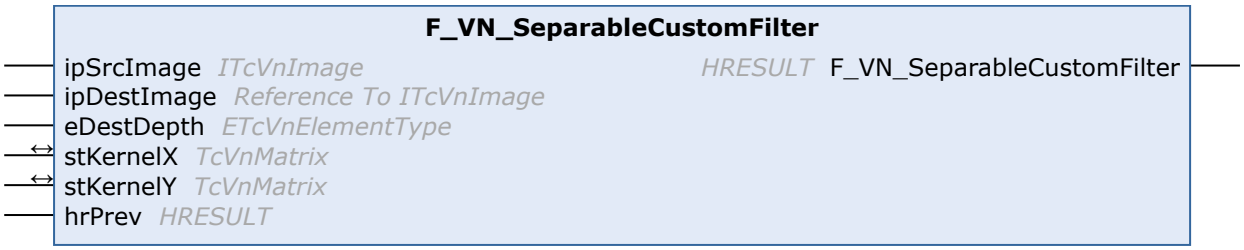
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.23 F_VN_SeparableCustomFilter



Apply a separable custom filter to the image.

Syntax

Definition:

```
FUNCTION F_VN_SeparableCustomFilter : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    eDestDepth  : ETcVnElementType;
END_VAR
VAR_IN_OUT
    stKernelX   : TcVnMatrix;
    stKernelY   : TcVnMatrix;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 210]	Destination image depth
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stKernelX	TcVnMatrix [▶ 245]	1D custom row-filter kernel with values of type REAL or LREAL
stKernelY	TcVnMatrix [▶ 245]	1D custom column-filter kernel with values of type REAL or LREAL

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_SeparableCustomFilter` wendet einen trennbaren 2D-Filter auf das Eingangsbild an.

Algorithmus

Ein trennbarer 2D Filter kann aus zwei 1D Filtern kombiniert werden. Es ist effizienter 2-mal einen 1D Filter auf ein Bild anzuwenden als 1-mal einen 2D Filter. So kann z. B. der bekannte Sobel-Filter [▶ 1350] wie folgt aus 2 separaten eindimensionalen Filtern aufgebaut werden:

$$stKernelX = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \qquad stKernelY = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

$$G = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \times \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` darf jedes verfügbare Format haben.

Ergebnisbild

Das Ergebnisbild `ipDestImage` hat die gleiche Größe wie das Eingangsbild `ipSrcImage`, das Pixelformat wird allerdings durch die Ergebnistiefe `eDestDepth` bestimmt.

X- und Y-Kernel

Die beiden 1-dimensionalen Filter-Kernel `stKernelX` und `stKernelY` werden als Matrix übergeben. Erlaubte Element-Typen der Matrix sind `TCVN_ET_REAL` und `TCVN_ET_LREAL`. Entsprechende Matrizen vom Typ TcVnMatrix [▶ 245] können mit Hilfe der Funktion F_VN_InitMatrixStruct [▶ 1570] erstellt werden.

Expert-Parameter

Die Expert-Variante F_VN_SeparableCustomFilterExp [▶ 1348] enthält zusätzliche Parameter.

Anwendung

Die Erstellung und Anwendung eines trennbaren 2D-Filters sieht z. B. so aus:

```
hr := F_VN_SeparableCustomFilter(
    ipSrcImage  :=  ipImageIn,
    ipDestImage :=  ipImageRes,
    eDestDepth  :=  TCVN_ET_USINT,
    stKernelX   :=  stKernelMatrixX,
    stKernelY   :=  stKernelMatrixY,
    hrPrev      :=  hr
);
```

Beispiele

- Benutzerdefinierte Filter

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.24 F_VN_SeparableCustomFilterExp

Apply a separable custom filter to the image. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_SeparableCustomFilterExp : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    eDestDepth  : ETcVnElementType;
END_VAR
VAR_IN_OUT
    stKernelX   : TcVnMatrix;
    stKernelY   : TcVnMatrix;
    aAnchor     : TcVnPoint;
END_VAR
VAR_INPUT
    fDelta      : LREAL;
    eBorderType : ETcVnBorderInterpolationMethod;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 210]	Destination image depth
fDelta	LREAL	Value added to each pixel after filtering
eBorderType	ETcVnBorderInterpolation Method [▶ 177]	Image border handling
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stKernelX	TcVnMatrix [▶ 245]	1D custom row-filter kernel with values of type REAL or LREAL
stKernelY	TcVnMatrix [▶ 245]	1D custom column-filter kernel with values of type REAL or LREAL
aAnchor	TcVnPoint [▶ 171]	Anchor point of the kernel

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_SeparableCustomFilterExp` ist die Expert-Variante von F_VN_SeparableCustomFilter [▶ 1346]. Sie enthält zusätzliche Parameter.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` darf jedes verfügbare Format haben.

Ergebnisbild

Das Ergebnisbild `ipDestImage` hat die gleiche Größe wie das Eingangsbild `ipSrcImage`, das Pixelformat wird allerdings durch die Ergebnistiefe `eDestDepth` bestimmt.

X- und Y-Kernel

Die beiden 1-dimensionalen Filter-Kernel `stKernelX` und `stKernelY` werden als Matrix übergeben. Erlaubte Element-Typen der Matrix sind `TCVN_ET_REAL` und `TCVN_ET_LREAL`. Entsprechende Matrizen vom Typ TcVnMatrix [▶ 245] können mit Hilfe der Funktion F_VN_InitMatrixStruct [▶ 1570] erstellt werden.

Ankerpunkt

Der Ankerpunkt `aAnchor` definiert, welcher Punkt der Filter-Kernel über jedes Pixel des Eingangsbildes `ipSrcImage` bewegt wird.

Delta

Das konstante Delta `fDelta` wird auf das Ergebnis der Filter-Operation addiert.

Rand-Extrapolation

Die Art der Rand-Extrapolation `eBorderType` definiert, wie nichtexistente Pixel über die Bildgrenze hinaus extrapoliert werden, um die Filterwerte an den Bildgrenzen auszurechnen. Für weitere Details siehe ETcVnBorderInterpolationMethod [▶ 177].

Anwendung

```
hr := F_VN_SeparableCustomFilterExp(
    ipSrcImage  :=  ipImageIn,
    ipDestImage :=  ipImageRes,
    eDestDepth  :=  TCVN_ET_USINT,
    stKernelX   :=  stKernelMatrixX,
    stKernelY   :=  stKernelMatrixY,
    aAnchor     :=  aAnchorPoint,
    fDelta      :=  0,
    eBorderType :=  TCVN_BIM_DEFAULT,
    hrPrev      :=  hr
);
```

Beispiele

- Benutzerdefinierte Filter

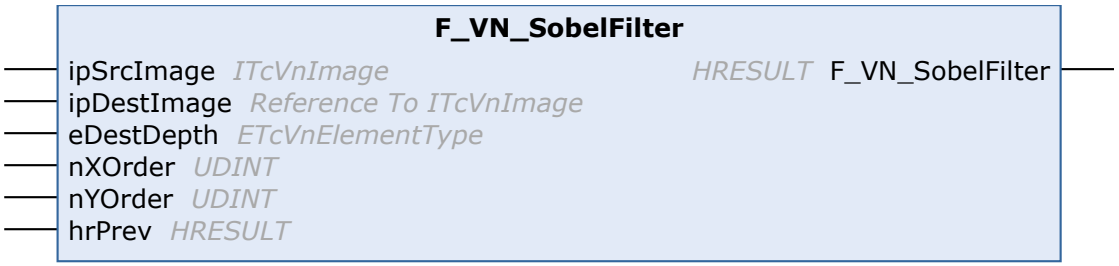
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.25 F_VN_SobelFilter



Calculates the first, second, or mixed image derivatives using an extended Sobel filter.

Syntax

Definition:

```
FUNCTION F_VN_SobelFilter : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    eDestDepth  : ETcVnElementType;
    nXOrder     : UDINT;
    nYOrder     : UDINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 210]	Destination image depth
nXOrder	UDINT	Order of the x-derivative (0, 1, 2)
nYOrder	UDINT	Order of the y-derivative (0, 1, 2)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_SobelFilter` wendet einen Sobel-Filter auf das Eingangsbild an. Der Sobel-Filter dient zur Kantendetektion. Kanten im Eingangsbild werden als helle Linien auf dunklem Hintergrund im Ergebnisbild hervorgehoben.

Algorithmus

Der erweiterte Sobel-Filter kann verwendet werden um Kanten zu detektieren, indem die n-te Ableitung der Übergänge benachbarter Pixelintensitäten approximiert wird. Da die Ableitung auch negative Werte hat, empfiehlt es sich einen vorzeichenbehafteten Elementtyp für das Ergebnisbild zu verwenden (z. B. `TCVN_ET_INT`).

Der Kernel des Filters wird durch die Parameter der Funktion bestimmt. Typischerweise wird die Funktion mit nXOrder = 1, nYOrder = 0, nKernelSize = 3 oder nXOrder = 0, nYOrder = 1, nKernelSize = 3 aufgerufen. Diese Parameter führen zu den Kerneln G_x und G_y.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Für diese Kernel kombiniert der Filter die Gaußsche Glättung mit der Berechnung der ersten Ableitung, so dass das Ergebnis recht rauschresistent ist. G_x betont vertikale Kanten, G_y betont horizontale Kanten.

Parameter

Originalbild

Das Originalbild `ipSrcImage` kann jedes verfügbare Format haben.

Ergebnisbild

Das Ergebnisbild `ipDestImage` liefert das Filter-Ergebnis zurück und besitzt das gleiche Format wie das Originalbild `ipSrcImage`.

Ergebnistiefe

Die Ergebnistiefe `eDestDepth` definiert, welchen Element-Typ das Ergebnisbild `ipDestImage` hat. Ein größerer Element-Typ kann mehr Informationen darstellen.

Ableitungsordnung

Die Ableitungsordnungen `nXOrder` und `nYOrder` definieren, die wievielte Ableitung in X- und Y-Richtung verwendet werden soll, um den Filter zu erstellen.

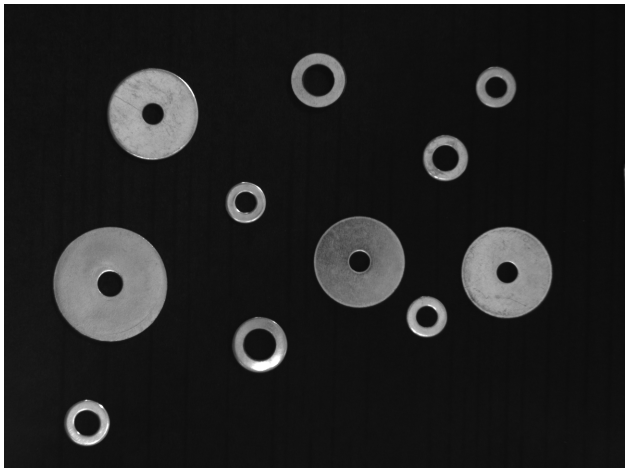
Expert-Parameter

Weitere Parameter finden Sie in der Expert-Variante F_VN_SobelFilterExp [▶ 1352].

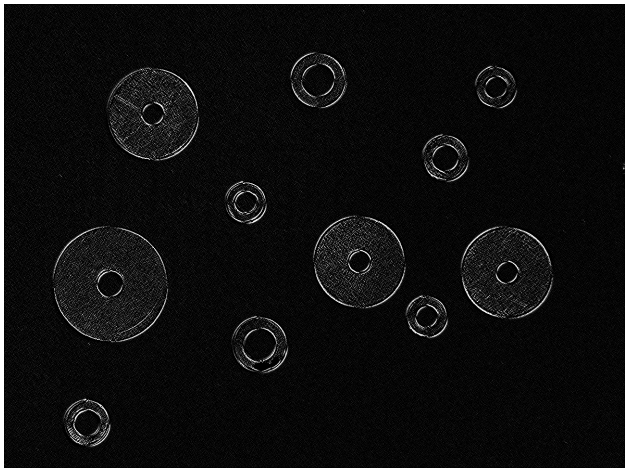
Anwendung

```
hr := F_VN_SobelFilter(
    ipSrcImage  :=  ipImageIn,
    ipDestImage :=  ipImageRes,
    eDestDepth  :=  TCVN_ET_USINT,
    nXOrder     :=  1,
    nYOrder     :=  1,
    hrPrev      :=  hr,
);
```

Originalbild



Ergebnisbild



Beispiele

- Kantendetektionsfilter [▶ 2894]

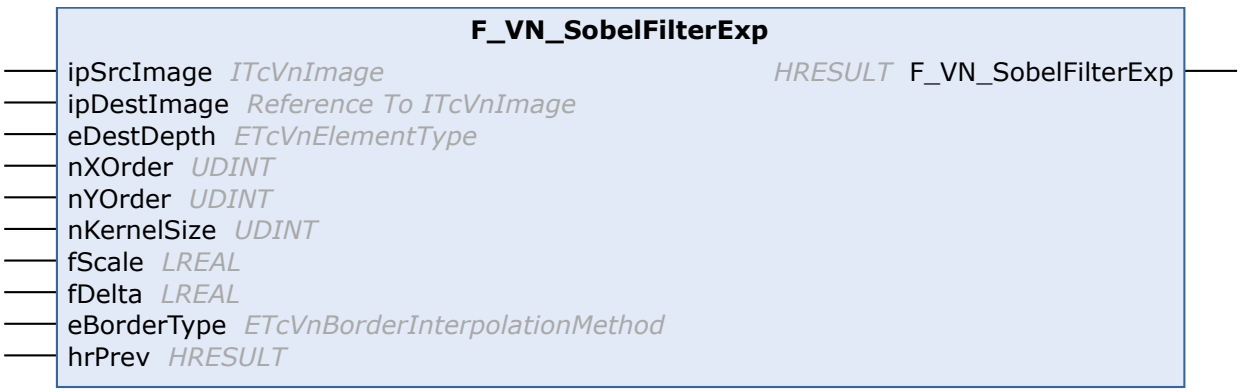
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.26 F_VN_SobelFilterExp



Calculates the first, second, third, or mixed image derivatives using an extended Sobel filter. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_SobelFilterExp : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    eDestDepth  : ETcVnElementType;
    nXOrder     : UDINT;
    nYOrder     : UDINT;
    nKernelSize : UDINT;
    fScale      : LREAL;
    fDelta      : LREAL;
    eBorderType : ETcVnBorderInterpolationMethod;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 210]	Destination image depth
nXOrder	UDINT	Order of the x-derivative (must be < nKernelSize)
nYOrder	UDINT	Order of the y-derivative (must be < nKernelSize)
nKernelSize	UDINT	Size of the extended Sobel kernel (3, 5, 7, ..., 31)
fScale	LREAL	Scale factor for the computed derivative values
fDelta	LREAL	Delta value that is added to the results prior to storing them in dest
eBorderType	ETcVnBorderInterpolationMethod [▶ 177]	Image border handling
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_SobelFilterExp` ist die Expert-Variante von F_VN_SobelFilter [▶ 1350]. Sie enthält zusätzliche Parameter.

Parameter

Originalbild

Das Originalbild `ipSrcImage` kann jedes verfügbare Format haben.

Ergebnisbild

Das Ergebnisbild `ipDestImage` liefert das Filter-Ergebnis zurück und besitzt das gleiche Format wie das Originalbild `ipSrcImage`.

Ergebnistiefe

Die Ergebnistiefe `eDestDepth` definiert, welchen Element-Typ das Ergebnisbild `ipDestImage` hat. Ein größerer Element-Typ kann mehr Informationen darstellen.

Ableitungsordnung

Die Ableitungsordnungen `nXOrder` und `nYOrder` definieren, die wievielte Ableitung in X- und Y-Richtung verwendet werden soll, um den Filter zu erstellen.

Kernel-Größe

Die Größe nKernelSize des Sobel-Kernels bestimmt, wie groß der Bereich ist, der den Ergebniswert eines Pixels beeinflusst.

Skalierung

Das Ergebnis der Sobel-Operation wird mit dem Skalierungsfaktor `fScale` multipliziert.

Delta

Das konstante Delta `fDelta` wird auf das Ergebnis der Sobel-Operation nach Skalierung mit `fScale` dazuaddiert.

Rand-Extrapolation

Die Art der Rand-Extrapolation `eBorderType` definiert, wie nichtexistente Pixel über die Bildgrenze hinaus extrapoliert werden, um die Filterwerte an den Bildgrenzen auszurechnen. Für weitere Details siehe ETcVnBorderInterpolationMethod [▶ 177].

Anwendung

```
hr := F_VN_SobelFilterExp(
    ipSrcImage  :=  ipImageIn,
    ipDestImage :=  ipImageRes,
    eDestDepth  :=  TCVN_ET_USINT,
    nXOrder     :=  1,
    nYOrder     :=  1,
    nKernelSize :=  3,
    fScale      :=  1,
    fDelta      :=  0,
    eBorderType :=  TCVN_BIM_DEFAULT,
    hrPrev      :=  hr,
);
```

Beispiele

- Kantendetektionsfilter [▶ 2894]

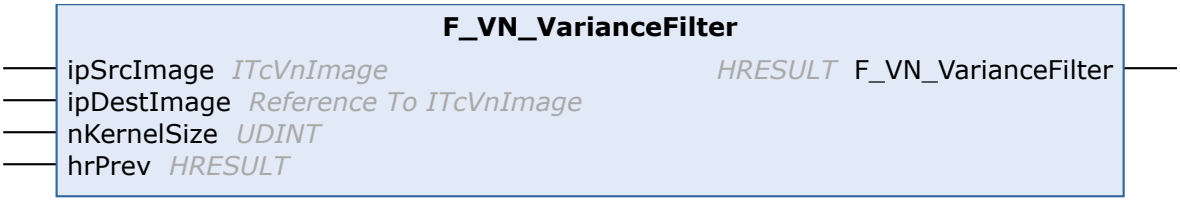
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.27 F_VN_VarianceFilter



Calculates the local variance of an image.

Syntax

Definition:

```
FUNCTION F_VN_VarianceFilter : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    nKernelSize : UDINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image. Only TCVN_ET_USINT Type is supported.
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image of type TCVN_ET_UINT (An appropriate destination image will be created if required.)
nKernelSize	UDINT	Size of the kernel
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

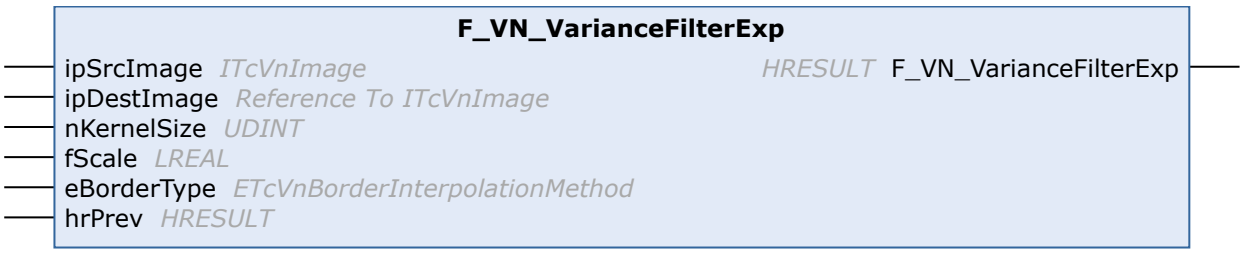
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.16.28 F_VN_VarianceFilterExp



Calculates the local variance of an image. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_VarianceFilterExp : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    nKernelSize : UDINT;
    fScale      : LREAL;
    eBorderType : ETcVnBorderInterpolationMethod;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image. Only TCVN_ET_USINT Type is supported.
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image of type TCVN_ET_UINT (An appropriate destination image will be created if required.)
nKernelSize	UDINT	Size of the kernel
fScale	LREAL	Scale factor for the computed variance values
eBorderType	ETcVnBorderInterpolationMethod [▶ 177]	Image border handling
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.17 Image Segmentation

Diese Gruppe enthält Funktionen zur Segmentierung von Bildern.

Funktionen

Segmentierung

- F_VN_AdaptiveThreshold(Exp) [▶ 1358]
- F_VN_CheckColorRange [▶ 1360]
- F_VN_DoubleThreshold [▶ 1361]
- F_VN_Threshold [▶ 1363]
- F_VN_WatershedSegmentationExp [▶ 1365]
- F_VN_SauvolaThreshold [▶ 1366]

Beispiele

Image Segmentation [▶ 2897]

6.1.4.17.1 F_VN_AdaptiveThreshold

Apply an adaptive threshold to a gray level image.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_AdaptiveThreshold : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    fMaxValue   : LREAL;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel, elements of type USINT)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
fMaxValue	LREAL	Value assigned to pixels for which the threshold condition is true
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_AdaptiveThreshold` wendet einen adaptiven Threshold auf das Eingangsbild an.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Grauwertbild mit einer Bittiefe von 8 Bit (`USINT`) sein.

Ergebnisbild

Das binäre Ergebnisbild `ipDestImage` hat das gleiche Format wie das Eingangsbild. Die Pixel haben entweder den Wert `0` oder den Zielwert `fMaxValue`.

Zielwert

Der Zielwert `fMaxValue` definiert, welchen Wert die Pixel bekommen sollen, die das Schwellwert-Kriterium erfüllen. Wenn das Ergebnisbild nur aus schwarzen und weißen Pixeln bestehen soll, muss hier der Maximal-Wert des Element-Typs des Eingangsbilds eingestellt werden. Dies entspricht `(2^d)-1`, wobei d die Bittiefe des Element-Typs darstellt; z. B. `(2^8)-1=255` bei 8-Bit-Bildern.

Expert-Parameter

Die Expert-Variante F_VN_AdaptiveThresholdExp [▶ 1358] enthält zusätzliche Parameter.

Anwendung

```
hr := F_VN_AdaptiveThreshold(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    fMaxValue       :=  255, // max value of 8-bit image: (2^8)-1=255
    hrPrev          :=  hr
);
```

Verwandte Funktionen

- F_VN_DoubleThreshold [▶ 1361] für binäre Segmentierung anhand von Doppelschwellwerten mittels morphologischer Rekonstruktion
- F_VN_Threshold [▶ 1363] für binäre Segmentierung anhand eines Schwellwerts
- F_VN_AdaptiveThreshold [▶ 1356] für binäre, adaptive Segmentierung
- F_VN_CheckColorRange [▶ 1360] für binäre Segmentierung anhand eines Farbspektrums
- F_VN_ReferenceColorSimilarity [▶ 1283] für binäre Segmentierung anhand eines Farbmodells

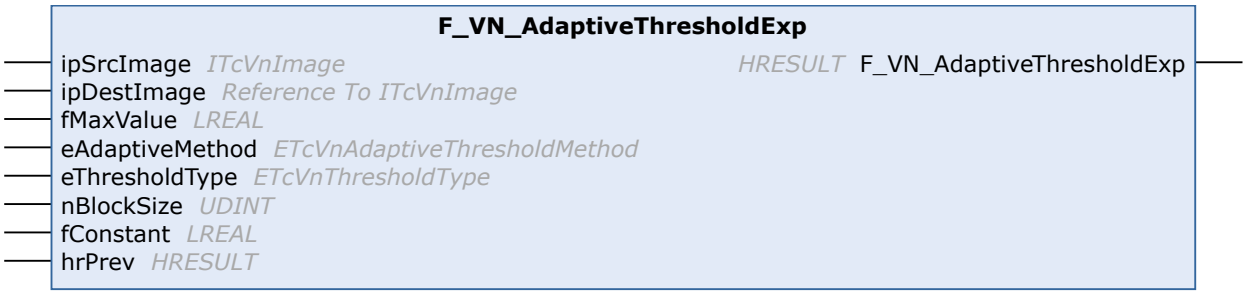
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.17.2 F_VN_AdaptiveThresholdExp



Apply an adaptive threshold to a gray level image. (expert function)
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_AdaptiveThresholdExp : HRESULT
VAR_INPUT
    ipSrcImage      : ITcVnImage;
    ipDestImage     : Reference To ITcVnImage;
    fMaxValue       : LREAL;
    eAdaptiveMethod : ETcVnAdaptiveThresholdMethod;
    eThresholdType  : ETcVnThresholdType;
    nBlockSize      : UDINT;
    fConstant       : LREAL;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel, elements of type USINT)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
fMaxValue	LREAL	Value assigned to pixels for which the threshold condition is true
eAdaptiveMethod	ETcVnAdaptiveThresholdMethod [▶ 174]	Adaptive threshold method to be applied (MEAN: threshold value will be calculated as a mean of the nBlockSize x nBlockSize neighborhood of (x,y) minus fConstant. GAUSSIAN: threshold value is a weighted sum (cross-correlation with a Gaussian window) of the nBlockSize x nBlockSize neighborhood of (x,y) minus fConstant.
eThresholdType	ETcVnThresholdType [▶ 232]	Threshold type to be applied (only BINARY and BINARY_INV are supported)
nBlockSize	UDINT	Size of the pixel neighborhood to calculate the local threshold (3, 5, 7, ...)
fConstant	LREAL	Constant that is subtracted from the weighted mean of the pixel neighborhood, which leads to the local threshold
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_AdaptiveThresholdExp` ist die Expert-Variante von F_VN_AdaptiveThreshold [▶ 1356]. Sie enthält zusätzliche Parameter.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Grauwertbild mit einer Bittiefe von 8 Bit (`USINT`) sein.

Ergebnisbild

Das binäre Ergebnisbild `ipDestImage` hat das gleiche Format wie das Eingangsbild. Die Pixel haben entweder den Wert `0` oder den Zielwert `fMaxValue`.

Zielwert

Der Zielwert `fMaxValue` definiert, welchen Wert die Pixel bekommen sollen, die das Schwellwert-Kriterium erfüllen. Wenn das Ergebnisbild nur aus schwarzen und weißen Pixeln bestehen soll, muss hier der Maximal-Wert des Element-Typs des Eingangsbilds eingestellt werden. Dies entspricht `(2^d)-1`, wobei d die Bittiefe des Element-Typs darstellt; z. B. `(2^8)-1=255` bei 8-Bit-Bildern.

Schwellwert-Methode

Die Schwellwert-Methode eAdaptiveMethod legt fest, ob der adaptive Schwellwert als Durchschnitt oder als Gauss-gewichtete Summe berechnet wird. Dazu stehen in dem Enum ETcVnAdaptiveThresholdMethod [▶ 174] die beiden Werte `TCVN_ATM_MEAN` und `TCVN_ATM_GAUSSIAN` zur Verfügung.

Schwellwert-Invertierung

Der Parameter eThresholdType legt fest, ob das Ergebnisbild invertiert werden soll oder nicht. Daher werden an dieser Stelle aus dem Enum ETcVnThresholdType [▶ 232] nur die beiden Werte `TCVN_TT_BINARY` und `TCVN_TT_BINARY_INV` unterstützt.

Schwellwert-Nachbarschaftsgröße

Der Parameter `nBlockSize` legt die Größe der Nachbarschaft fest, die für die Berechnung des lokalen Schwellwerts betrachtet wird. Der Parameter muss ungerade sein (3, 5, 7, …).

Schwellwert-Anpassung

Der konstante Wert `fConstant` wird von jedem lokalen Durchschnittswert abgezogen. Das Ergebnis ist der lokale Schwellwert, der zur Berechnung des Ergebnisbildes verwendet wird.

Anwendung

```
hr := F_VN_AdaptiveThreshold(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    fMaxValue       :=  255, // max value of 8-bit image: (2^8)-1=255
    eAdaptiveMethod :=  TCVN_ATM_MEAN,
    eThresholdType  :=  TCVN_TT_BINARY_INV,
    nBlockSize      :=  5,
    fConstant       :=  10,
    hrPrev          :=  hr
);
```

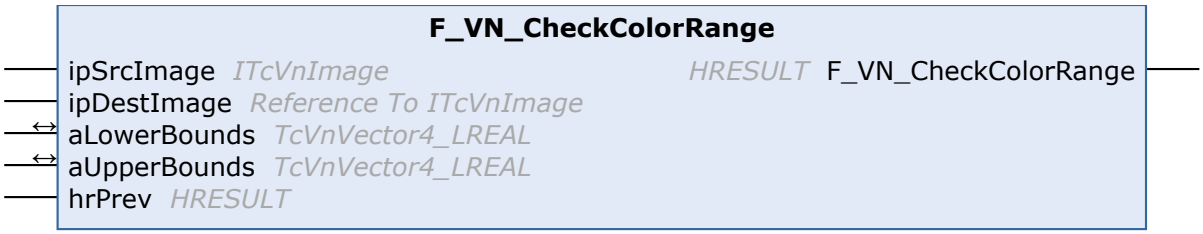
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.17.3 F_VN_CheckColorRange



Check if the pixel values of an image lie in a given range. The destination image has the same size as the source image but only one channel with a pixel element size of 8 bit. Its elements are set to 255 if the corresponding pixels of the source image are in the checked range and set to 0 otherwise.

Syntax

Definition:

```
FUNCTION F_VN_CheckColorRange : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipDestImage   : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    aLowerBounds : TcVnVector4_LREAL;
    aUpperBounds : TcVnVector4_LREAL;
END_VAR
```

```
VAR_INPUT
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLowerBounds	TcVnVector4_LREAL [▶ 173]	Channel-wise lower bounds (Unused channels are ignored.)
aUpperBounds	TcVnVector4_LREAL [▶ 173]	Channel-wise upper bounds (Unused channels are ignored.)

Return value

HRESULT [▶ 150]

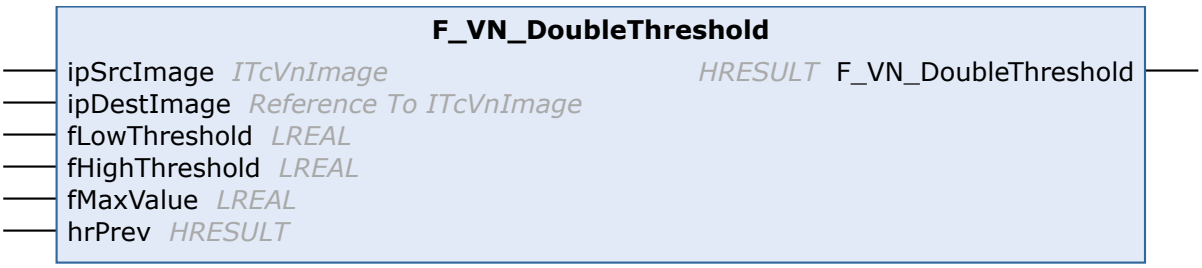
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.17.4 F_VN_DoubleThreshold



Apply a double threshold (also called hysteresis threshold) using morphological reconstruction. Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_DoubleThreshold : HRESULT
VAR_INPUT
    ipSrcImage     : ITcVnImage;
    ipDestImage    : Reference To ITcVnImage;
    fLowThreshold  : LREAL;
    fHighThreshold : LREAL;
```

```
    fMaxValue       : LREAL;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
fLowThreshold	LREAL	Low threshold determinining the shape of found regions
fHighThreshold	LREAL	High threshold selecting found regions (Each region must at least contain a single pixel if the high threshold is applied.)
fMaxValue	LREAL	Maximum pixel value
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_DoubleThreshold` berechnet einen Doppelschwellwert mittels morphologischer Rekonstruktion. Die Funktion erwartet, dass die gesuchten Segmente heller sind als der Hintergrund und ermöglicht, im Vergleich zur Funktion `F_VN_Threshold`, eine zusätzliche Filterung der ermittelten Bildsegmente.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Grauwertbild sein.

Ergebnisbild

Das binäre Ergebnisbild `ipDestImage` hat das gleiche Format wie das Eingangsbild. Die Pixel haben entweder den Wert `0` oder den Zielwert `fMaxValue`.

Unterer Schwellwert

Der untere Schwellwert bestimmt die Form der gefundenen Segmente wie die Funktion `F_VN_Threshold`.

Oberer Schwellwert

Der obere Schwellwert dient dazu, Bildsegmente zu markieren, deren Form dann auf der Basis des niedrigen Schwellwerts berechnet wird. Auf diese Weise können unerwünschte Segmente herausgefiltert werden, deren Grauwerte größer als die untere Schwelle und kleiner als die obere Schwelle sind.

Zielwert

Der Zielwert `fMaxValue` definiert, welchen Wert die Pixel bekommen sollen, die das jeweilige Schwellwert-Kriterium erfüllen. Wenn das Ergebnisbild nur aus schwarzen und weißen Pixeln bestehen soll, muss hier der Maximal-Wert des Element-Typs des Eingangsbilds eingestellt werden. Dies entspricht `(2^d)-1`, wobei d die Bittiefe des Element-Typs darstellt; z. B. `(2^8)-1=255` bei 8-Bit-Bildern.

Anwendung

```
hr := F_VN_DoubleThreshold(
    ipSrcImage      := ipSrcImage,
    ipDestImage     := ipDestImage,
    fLowThreshold   := fLowThreshold,
    fHighThreshold  := fHighThreshold,
    fMaxValue       := 255,
    hrPrev          := hr);
```

Verwandte Funktionen

- F_VN_DoubleThreshold [▶ 1361] für binäre Segmentierung anhand von Doppelschwellwerten mittels morphologischer Rekonstruktion
- F_VN_Threshold [▶ 1363] für binäre Segmentierung anhand eines Schwellwerts
- F_VN_AdaptiveThreshold [▶ 1356] für binäre, adaptive Segmentierung
- F_VN_CheckColorRange [▶ 1360] für binäre Segmentierung anhand eines Farbspektrums
- F_VN_ReferenceColorSimilarity [▶ 1283] für binäre Segmentierung anhand eines Farbmodells

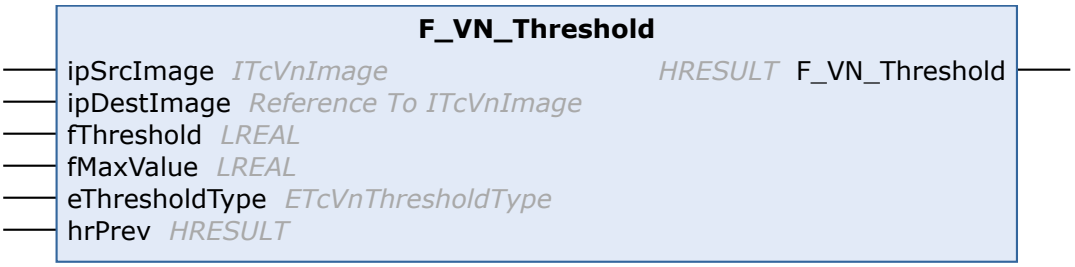
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.17.5 F_VN_Threshold



Apply a fixed threshold or a dynamic threshold according to Otsu.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_Threshold : HRESULT
VAR_INPUT
    ipSrcImage      : ITcVnImage;
    ipDestImage     : Reference To ITcVnImage;
    fThreshold      : LREAL;
    fMaxValue       : LREAL;
    eThresholdType  : ETcVnThresholdType;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (For Otsu and Triangle threshold types, only 1 channel of TCVN_ET_USINT is supported)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
fThreshold	LREAL	Fixed threshold (unused if dynamic thresholding is selected)
fMaxValue	LREAL	Maximum pixel value
eThresholdType	ETcVnThresholdType [▶ 232]	Threshold type to be applied
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_Threshold` segmentiert das Eingangsbild hinsichtlich eines Schwellwerts. Abhängig von diesem Schwellwert und dem verwendeten Algorithmus wird jedem Pixel des binären Ergebnisbilds der Wert `0` und ein vorgegebener Zielwert zugewiesen.

Algorithmus

Ein Threshold verwandelt ein monochromes Bild in ein binäres Bild und lässt so Segmente entstehen. Eine gute Segmentierung ist die Voraussetzung für viele Folgemethoden wie z. B. die Kontursuche. TwinCAT Vision bietet verschiedene Arten von Schwellwerten, die für unterschiedliche Anwendungen geeignet sind.

Fester Schwellwert

Ein fester Schwellwert erzeugt einfach ein binäres Bild, indem Pixel mit einem Wert größer als fThreshold auf weiß (255) und andere auf schwarz (0) gesetzt werden. fThreshold ist ein Parameter der entsprechenden Funktionen. Um einen guten Wert herauszufinden, werden Sie sich höchstwahrscheinlich durch verschiedene Versuche annähern.

Dynamischer Schwellwert

Eine dynamische Schwelle nach Otsu funktioniert perfekt für bimodale Bilder. Das Histogramm eines bimodalen Bildes hat zwei Hauptpeaks und ein guter Schwellwert liegt höchstwahrscheinlich genau in der Mitte. Falls Sie ein verrauschtes Bild haben, können Sie einen Gauß-Filter verwenden, um es in eine bimodale Richtung zu verschieben.

Adaptiver Schwellwert

Ein adaptiver Schwellwert verwendet variable Schwellwerte im gesamten Bild. Siehe F_VN_AdaptiveThresholdExp [▶ 1358] .

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` darf einen oder drei Kanäle besitzen.

Ergebnisbild

Das Ergebnisbild `ipDestImage` gibt das segmentierte Binärbild zurück.

Schwellwert

Der Schwellwert `fThreshold` definiert die Segmentierungsbedingung. Er muss je nach Beschaffenheit des Eingangsbilds angepasst werden.

Zielwert

Der Zielwert `fMaxValue` definiert, welchen Wert die Pixel bekommen sollen, die das Schwellwert-Kriterium erfüllen. Wenn das Ergebnisbild nur aus schwarzen und weißen Pixeln bestehen soll, muss hier der Maximal-Wert des Element-Typs des Eingangsbilds eingestellt werden. Dies entspricht `(2^d)-1`, wobei `d` die Bittiefe des Element-Typs darstellt; z. B. `(2^8)-1=255` bei 8-Bit-Bildern.

Schwellwert-Typ

Der Schwellwert-Typ `eThresholdType` definiert, nach welchem Algorithmus der Schwellwert `fThreshold` auf das Eingangsbild angewandt wird. Alle Werte des Enums ETcVnThresholdType [▶ 232] werden unterstützt.

Anwendung

Die binäre Segmentierung eines 8-Bit Grauwertbilds in der Mitte des Wertespektrums sieht z. B. so aus:

```
hr := F_VN_Threshold(
    ipSrcImage      :=  ipImageIn,
    ipDestImage     :=  ipImageRes,
    fThreshold      :=  128, // half value range: (2^8)/2=128
    fMaxValue       :=  255, // max value of 8-bit image: (2^8)-1=255
    eThresholdType  :=  TCVN_TT_BINARY,
    hrPrev          :=  hr
);
```

Verwandte Funktionen

- F_VN_DoubleThreshold [▶ 1361] für binäre Segmentierung anhand von Doppelschwellwerten mittels morphologischer Rekonstruktion
- F_VN_Threshold [▶ 1363] für binäre Segmentierung anhand eines Schwellwerts
- F_VN_AdaptiveThreshold [▶ 1356] für binäre, adaptive Segmentierung
- F_VN_CheckColorRange [▶ 1360] für binäre Segmentierung anhand eines Farbspektrums
- F_VN_ReferenceColorSimilarity [▶ 1283] für binäre Segmentierung anhand eines Farbmodells

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.17.6 F_VN_WatershedSegmentationExp

Apply a marker-based watershed segmentation. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_WatershedSegmentationExp : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    ipMarkers   : ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, 3 channels)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (DINT, 1 channel. An appropriate image will be created if required.)
ipMarkers	ITcVnImage [▶ 426]	Marker image (DINT, 1 channel)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.17.7 F_VN_SauvolaThreshold

Apply Sauvola local threshold to source image.

Syntax

Definition:

```
FUNCTION F_VN_SauvolaThreshold : HRESULT
VAR_INPUT
    ipSrcImage     : ITcVnImage;
    pipDestImage   : Reference To ITcVnImage;
    nBlockSize     : UDINT;
    fParamK        : LREAL;
    fMaxSigma      : LREAL;
```

```
    eThresholdType : ETcVnThresholdType;
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
pipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate single channel destination image will be created if required.))
nBlockSize	UDINT	Size of the pixel neighborhood to calculate the local threshold (3, 5, 7, ...)
fParamK	LREAL	parameter K which takes positive values in range (0,1). Recommended are K in range [0.2, 0.5]
fMaxSigma	LREAL	Maximum value of the standard deviation, required to be strictly positive. (fMaxSigma = 128 for a grayscale document).
eThresholdType	ETcVnThresholdType [▶ 232]	Threshold type to be applied. Supported are TCVN_TT_BINARY and TCVN_TT_BINARY_INV.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18 Keypoint Features

Diese Gruppe enthält Funktionen zum Finden und Matchen von markanten Features in einem Bild.

HINWEIS

Hohe Rechendauer

Viele Funktionen zum Finden von Keypoint Features benötigen eine hohe Rechendauer. Achten Sie daher auf die einstellte Zykluszeit und verwenden Sie ggfs. einen Watchdog [▶ 156] zur Absicherung des Echtzeitverhaltens.

Funktionen

Key Points

- F_VN_KeyPointsAGAST(Exp) [▶ 1386]
- F_VN_KeyPointsFAST(Exp) [▶ 1395]
- F_VN_KeyPointsGFTT(Exp) [▶ 1397]
- F_VN_KeyPointsMSER(Exp) [▶ 1399]
- F_VN_KeyPointsSB(Exp) [▶ 1401]

Key Points und Deskriptoren

- F_VN_KeyPointsAndDescriptorsAKAZE(Exp) [▶ 1388]
- F_VN_KeyPointsAndDescriptorsBRISK(Exp) [▶ 1389]
- F_VN_KeyPointsAndDescriptorsaKAZE(Exp) [▶ 1391]
- F_VN_KeyPointsAndDescriptorsORB(Exp) [▶ 1393]

Deskriptor-Matching

- F_VN_FilterGoodMatches [▶ 1368]
- F_VN_GetMatchCoordinates [▶ 1385]
- F_VN_MatchDescriptorsBF(Exp) [▶ 1403]
- F_VN_MatchDescriptorsFlannLsh(Exp) [▶ 1405]
- F_VN_MatchDescriptorsKnnBF(Exp) [▶ 1407]
- F_VN_MatchDescriptorsKnnFlannLsh(Exp) [▶ 1409]

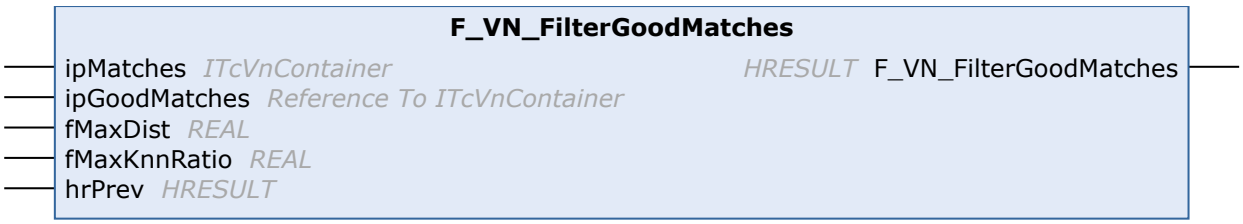
Key Points Referenz Matching

- F_VN_FindReferenceKeyPointsInImage(Exp) [▶ 1369]
- F_VN_FindReferenceKeyPointsInImageAKAZE(Exp) [▶ 1371]
- F_VN_FindReferenceKeyPointsInImageBRISK(Exp) [▶ 1375]
- F_VN_FindReferenceKeyPointsInImageORB(Exp) [▶ 1381]

Sonstiges

- F_VN_RegionsMSER(Exp) [▶ 1411]

6.1.4.18.1 F_VN_FilterGoodMatches



Filter the descriptor matches and return only good ones.

Syntax

Definition:

```
FUNCTION F_VN_FilterGoodMatches : HRESULT
VAR_INPUT
    ipMatches     : ITcVnContainer;
    ipGoodMatches : Reference To ITcVnContainer;
    fMaxDist      : REAL;
    fMaxKnnRatio  : REAL;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMatches	ITcVnContainer [▶ 385]	Container with descriptor matches (ContainerType_Vector_TcVnDMatch or ContainerType_Vector_Vector_TcVnDMatch)
ipGoodMatches	Reference To ITcVnContainer [▶ 385]	Returns a container with good matches (ContainerType_Vector_TcVnDMatch)
fMaxDist	REAL	Maximum allowed descriptor distance (-1 disables this filter criterion)
fMaxKnnRatio	REAL	Maximum allowed distance ratio between first and second best match ([0..1], used for knn match results only, -1 disables this filter criterion)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

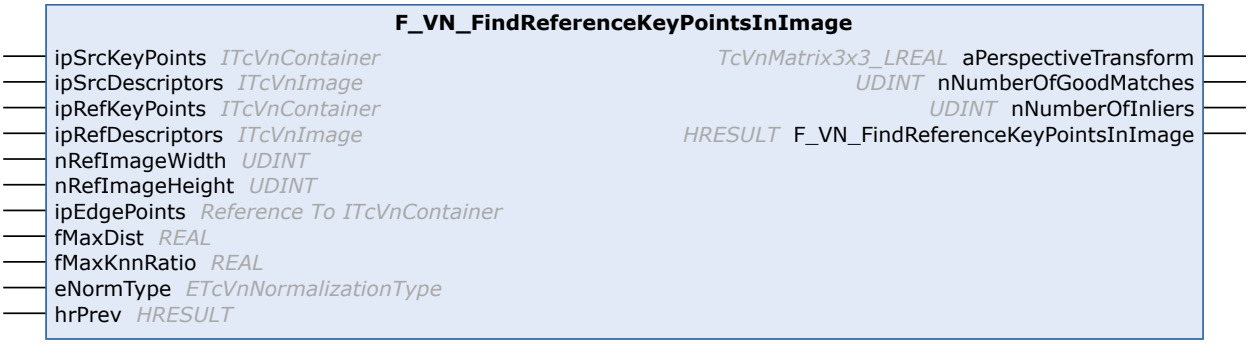
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.2 F_VN_FindReferenceKeyPointsInImage



Searches a reference image, represented by its keypoints and corresponding descriptors, in a given source image.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_FindReferenceKeyPointsInImage : HRESULT
VAR_INPUT
    ipSrcKeyPoints        : ITcVnContainer;
    ipSrcDescriptors      : ITcVnImage;
    ipRefKeyPoints        : ITcVnContainer;
    ipRefDescriptors      : ITcVnImage;
    nRefImageWidth        : UDINT;
    nRefImageHeight       : UDINT;
    ipEdgePoints          : Reference To ITcVnContainer;
    fMaxDist              : REAL;
```

```
    fMaxKnnRatio          : REAL;
    eNormType             : ETcVnNormalizationType;
    hrPrev                : HRESULT;
END_VAR
VAR_OUTPUT
    aPerspectiveTransform : TcVnMatrix3x3_LREAL;
    nNumberOfGoodMatches  : UDINT;
    nNumberOfInliers      : UDINT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcKeyPoints	ITcVnContainer [▶ 385]	KeyPoints of the source image, e.g. (ContainerType_Vector_TcVnKeyPoint)
ipSrcDescriptors	ITcVnImage [▶ 426]	Descriptors of the source image KeyPoints
ipRefKeyPoints	ITcVnContainer [▶ 385]	KeyPoints of the reference image, e.g. (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage [▶ 426]	Descriptors of the reference image KeyPoints
nRefImageWidth	UDINT	Width of the reference image in pixels
nRefImageHeight	UDINT	Height of the reference image in pixels
ipEdgePoints	Reference To ITcVnContainer [▶ 385]	Returns a container with the 4 edge points of the reference image transformed into the coordinates of the source image (ContainerType_Vector_TcVnPoint2_REAL)
fMaxDist	REAL	Maximum allowed descriptor distance (-1 disables this filter criterion)
fMaxKnnRatio	REAL	Maximum allowed distance ratio between first and second best match ([0..1], -1 disables this filter criterion)
eNormType	ETcVnNormalizationType [▶ 221]	Normalization type used for descriptor matching (HAMMING recommended for AKAZE, ORB, and BRISK. HAMMING2 is recommended for ORB if the ORB nBriefPoints parameter is 3 or 4.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Outputs

Name	Type	Description
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 173]	Returns the perspective transformation matrix, which transforms the reference points to the source points
nNumberOfGoodMatches	UDINT	Return the number of good matches
nNumberOfInliers	UDINT	Return the number of inlier matches

Return value

HRESULT [▶ 150]

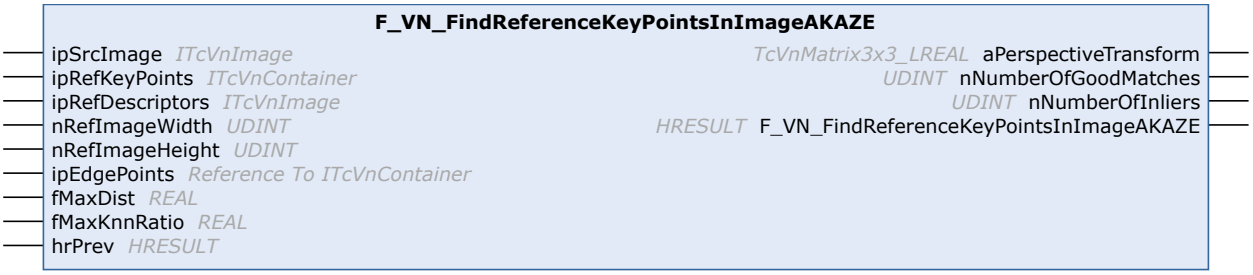
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.3 F_VN_FindReferenceKeyPointsInImageAKAZE



Searches a reference image, represented by its AKAZE keypoints and corresponding descriptors, in a given source image (uses the default TcVnParamsAKAZE parameters).
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_FindReferenceKeyPointsInImageAKAZE : HRESULT
VAR_INPUT
    ipSrcImage             : ITcVnImage;
    ipRefKeyPoints         : ITcVnContainer;
    ipRefDescriptors       : ITcVnImage;
    nRefImageWidth         : UDINT;
    nRefImageHeight        : UDINT;
    ipEdgePoints           : Reference To ITcVnContainer;
    fMaxDist               : REAL;
    fMaxKnnRatio           : REAL;
    hrPrev                 : HRESULT;
END_VAR
VAR_OUTPUT
    aPerspectiveTransform : TcVnMatrix3x3_LREAL;
    nNumberOfGoodMatches  : UDINT;
    nNumberOfInliers      : UDINT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipRefKeyPoints	ITcVnContainer [▶ 385]	KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsAKAZE (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage [▶ 426]	Descriptors of the reference image KeyPoints, e.g. computed with F_VN_KeyPointsAndDescriptorsAKAZE
nRefImageWidth	UDINT	Width of the reference image in pixels
nRefImageHeight	UDINT	Height of the reference image in pixels
ipEdgePoints	Reference To ITcVnContainer [▶ 385]	Returns a container with the 4 edge points of the reference image transformed into ipSrcImage coordinates (ContainerType_Vector_TcVnPoint2_REAL)
fMaxDist	REAL	Maximum allowed descriptor distance (-1 disables this filter criterion)
fMaxKnnRatio	REAL	Maximum allowed distance ratio between first and second best match ([0..1], -1 disables this filter criterion)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Outputs

Name	Type	Description
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 173]	Returns the perspective transformation matrix, which transforms the reference points to the source points
nNumberOfGoodMatches	UDINT	Return the number of good matches
nNumberOfInliers	UDINT	Return the number of inlier matches

Return value

HRESULT [▶ 150]

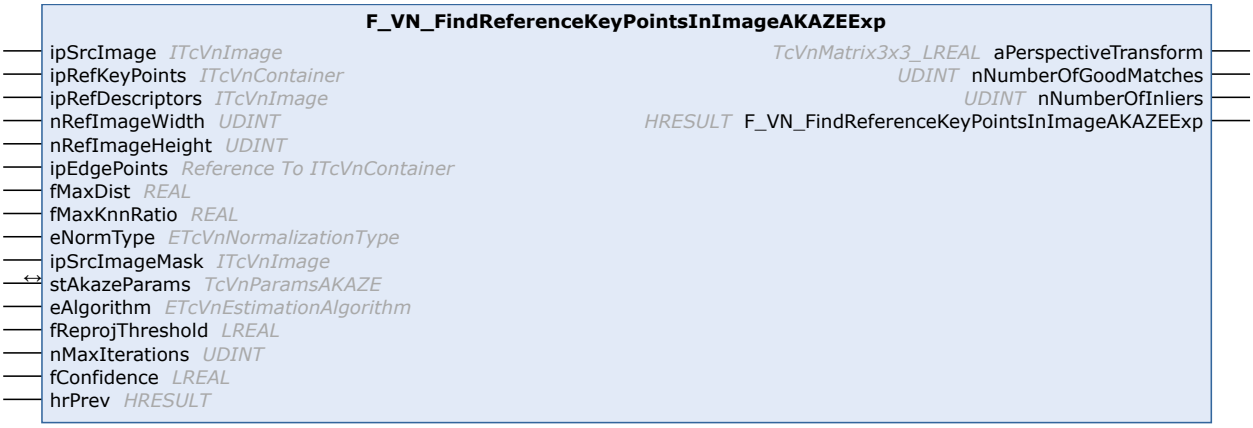
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.4 F_VN_FindReferenceKeyPointsInImageAKAZEExp



Searches a reference image, represented by its AKAZE keypoints and corresponding descriptors, in a given source image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_FindReferenceKeyPointsInImageAKAZEExp : HRESULT
VAR_INPUT
    ipSrcImage            : ITcVnImage;
    ipRefKeyPoints        : ITcVnContainer;
    ipRefDescriptors      : ITcVnImage;
    nRefImageWidth        : UDINT;
    nRefImageHeight       : UDINT;
    ipEdgePoints          : Reference To ITcVnContainer;
    fMaxDist              : REAL;
    fMaxKnnRatio          : REAL;
    eNormType             : ETcVnNormalizationType;
    ipSrcImageMask        : ITcVnImage;
END_VAR
VAR_IN_OUT
    stAkazeParams         : TcVnParamsAKAZE;
END_VAR
VAR_INPUT
    eAlgorithm            : ETcVnEstimationAlgorithm;
    fReprojThreshold      : LREAL;
    nMaxIterations        : UDINT;
    fConfidence           : LREAL;
    hrPrev                : HRESULT;
END_VAR
VAR_OUTPUT
    aPerspectiveTransform : TcVnMatrix3x3_LREAL;
    nNumberOfGoodMatches  : UDINT;
    nNumberOfInliers      : UDINT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipRefKeyPoints	ITcVnContainer [▶ 385]	KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsAKAZE (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage [▶ 426]	Descriptors of the reference image KeyPoints, e.g. computed with F_VN_KeyPointsAndDescriptorsAKAZE
nRefImageWidth	UDINT	Width of the reference image in pixels
nRefImageHeight	UDINT	Height of the reference image in pixels
ipEdgePoints	Reference To ITcVnContainer [▶ 385]	Returns a container with the 4 edge points of the reference image transformed into ipSrcImage coordinates (ContainerType_Vector_TcVnPoint2_REAL)
fMaxDist	REAL	Maximum allowed descriptor distance (-1 disables this filter criterion)
fMaxKnnRatio	REAL	Maximum allowed distance ratio between first and second best match ([0..1], -1 disables this filter criterion)
eNormType	ETcVnNormalizationType [▶ 221]	Normalization type used for descriptor matching
ipSrcImageMask	ITcVnImage [▶ 426]	Mask to specify, where to look for keypoints in ipSrcImage (set to 0 if no mask required)
eAlgorithm	ETcVnEstimationAlgorithm [▶ 211]	Estimation algorithm used for computing the perspective transformation between the two point sets
fReprojThreshold	LREAL	Maximum allowed reprojection error to treat a point pair as an inlier (only used if eAlgorithm is RANSAC, RHO)
nMaxIterations	UDINT	Maximum number of RANSAC iterations
fConfidence	LREAL	Confidence (0..1)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stAkazeParams	TcVnParamsAKAZE [▶ 247]	Parameters to configure the keypoint and descriptor computation for ipSrcImage (resulting descriptors must be compatible to ipRefDescriptors!)

Outputs

Name	Type	Description
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 173]	Returns the perspective transformation matrix, which transforms the reference points to the source points
nNumberOfGoodMatches	UDINT	Return the number of good matches
nNumberOfInliers	UDINT	Return the number of inlier matches

Return value

HRESULT [▶ 150]

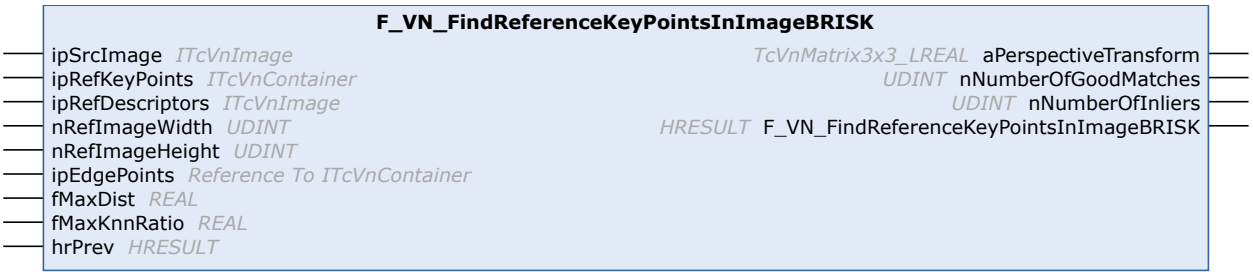
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.5 F_VN_FindReferenceKeyPointsInImageBRISK



Searches a reference image, represented by its BRISK keypoints and corresponding descriptors, in a given source image (uses the default TcVnParamsBRISK parameters).
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_FindReferenceKeyPointsInImageBRISK : HRESULT
VAR_INPUT
    ipSrcImage            : ITcVnImage;
    ipRefKeyPoints        : ITcVnContainer;
    ipRefDescriptors      : ITcVnImage;
    nRefImageWidth        : UDINT;
    nRefImageHeight       : UDINT;
    ipEdgePoints          : Reference To ITcVnContainer;
    fMaxDist              : REAL;
    fMaxKnnRatio          : REAL;
    hrPrev                : HRESULT;
END_VAR
VAR_OUTPUT
    aPerspectiveTransform : TcVnMatrix3x3_LREAL;
    nNumberOfGoodMatches  : UDINT;
    nNumberOfInliers      : UDINT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipRefKeyPoints	ITcVnContainer [▶ 385]	KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsBRISK (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage [▶ 426]	Descriptors of the reference image KeyPoints, e.g. computed with F_VN_KeyPointsAndDescriptorsBRISK
nRefImageWidth	UDINT	Width of the reference image in pixels
nRefImageHeight	UDINT	Height of the reference image in pixels
ipEdgePoints	Reference To ITcVnContainer [▶ 385]	Returns a container with the 4 edge points of the reference image transformed into ipSrcImage coordinates (ContainerType_Vector_TcVnPoint2_REAL)
fMaxDist	REAL	Maximum allowed descriptor distance (-1 disables this filter criterion)
fMaxKnnRatio	REAL	Maximum allowed distance ratio between first and second best match ([0..1], -1 disables this filter criterion)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Outputs

Name	Type	Description
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 173]	Returns the perspective transformation matrix, which transforms the reference points to the source points
nNumberOfGoodMatches	UDINT	Return the number of good matches
nNumberOfInliers	UDINT	Return the number of inlier matches

Return value

HRESULT [▶ 150]

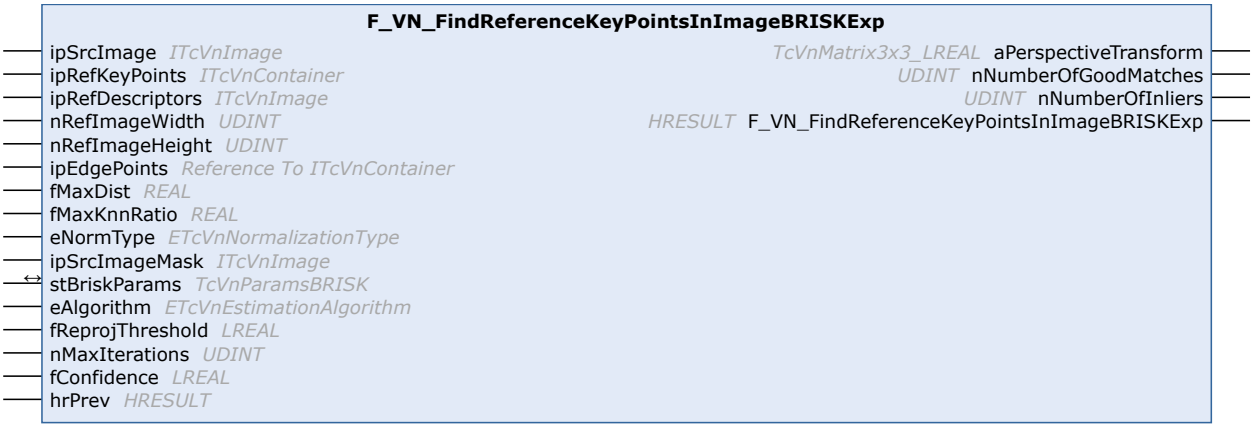
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.6 F_VN_FindReferenceKeyPointsInImageBRISKExp



Searches a reference image, represented by its BRISK keypoints and corresponding descriptors, in a given source image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_FindReferenceKeyPointsInImageBRISKExp : HRESULT
VAR_INPUT
    ipSrcImage            : ITcVnImage;
    ipRefKeyPoints        : ITcVnContainer;
    ipRefDescriptors      : ITcVnImage;
    nRefImageWidth        : UDINT;
    nRefImageHeight       : UDINT;
    ipEdgePoints          : Reference To ITcVnContainer;
    fMaxDist              : REAL;
    fMaxKnnRatio          : REAL;
    eNormType             : ETcVnNormalizationType;
    ipSrcImageMask        : ITcVnImage;
END_VAR
VAR_IN_OUT
    stBriskParams         : TcVnParamsBRISK;
END_VAR
VAR_INPUT
    eAlgorithm            : ETcVnEstimationAlgorithm;
    fReprojThreshold      : LREAL;
    nMaxIterations        : UDINT;
    fConfidence           : LREAL;
    hrPrev                : HRESULT;
END_VAR
VAR_OUTPUT
    aPerspectiveTransform : TcVnMatrix3x3_LREAL;
    nNumberOfGoodMatches  : UDINT;
    nNumberOfInliers      : UDINT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipRefKeyPoints	ITcVnContainer [▶ 385]	KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsBRISK (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage [▶ 426]	Descriptors of the reference image KeyPoints, e.g. computed with F_VN_KeyPointsAndDescriptorsBRISK
nRefImageWidth	UDINT	Width of the reference image in pixels
nRefImageHeight	UDINT	Height of the reference image in pixels
ipEdgePoints	Reference To ITcVnContainer [▶ 385]	Returns a container with the 4 edge points of the reference image transformed into ipSrcImage coordinates (ContainerType_Vector_TcVnPoint2_REAL)
fMaxDist	REAL	Maximum allowed descriptor distance (-1 disables this filter criterion)
fMaxKnnRatio	REAL	Maximum allowed distance ratio between first and second best match ([0..1], -1 disables this filter criterion)
eNormType	ETcVnNormalizationType [▶ 221]	Normalization type used for descriptor matching
ipSrcImageMask	ITcVnImage [▶ 426]	Mask to specify, where to look for keypoints in ipSrcImage (set to 0 if no mask required)
eAlgorithm	ETcVnEstimationAlgorithm [▶ 211]	Estimation algorithm used for computing the perspective transformation between the two point sets
fReprojThreshold	LREAL	Maximum allowed reprojection error to treat a point pair as an inlier (only used if eAlgorithm is RANSAC, RHO)
nMaxIterations	UDINT	Maximum number of RANSAC iterations
fConfidence	LREAL	Confidence (0..1)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stBriskParams	TcVnParamsBRISK [▶ 251]	Parameters to configure the keypoint and descriptor computation for ipSrcImage (resulting descriptors must be compatible to ipRefDescriptors!)

Outputs

Name	Type	Description
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 173]	Returns the perspective transformation matrix, which transforms the reference points to the source points
nNumberOfGoodMatches	UDINT	Return the number of good matches
nNumberOfInliers	UDINT	Return the number of inlier matches

Return value

HRESULT [▶ 150]

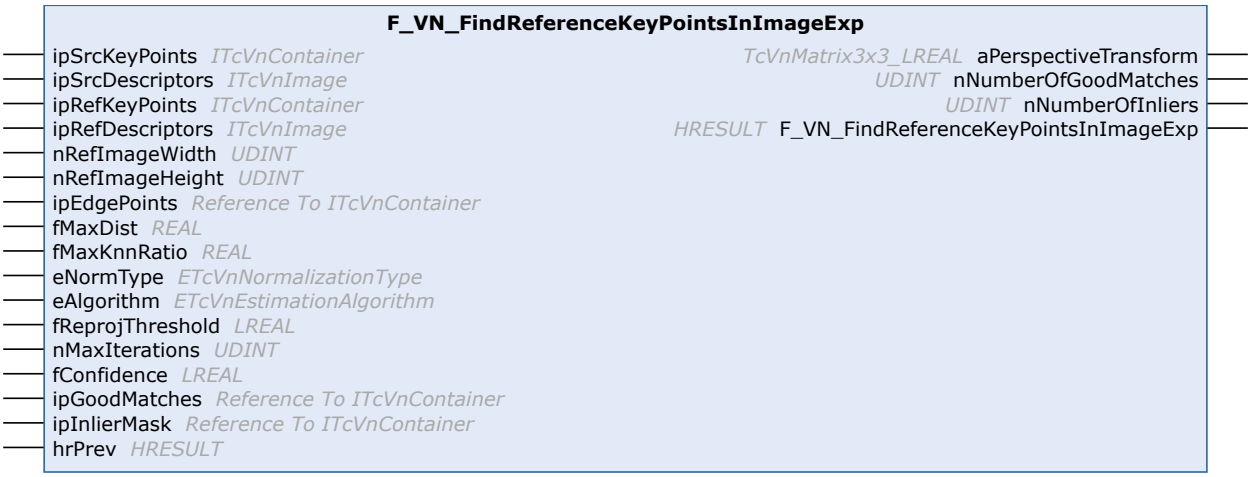
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.7 F_VN_FindReferenceKeyPointsInImageExp



Searches a reference image, represented by its keypoints and corresponding descriptors, in a given source image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_FindReferenceKeyPointsInImageExp : HRESULT
VAR_INPUT
    ipSrcKeyPoints        : ITcVnContainer;
    ipSrcDescriptors      : ITcVnImage;
    ipRefKeyPoints        : ITcVnContainer;
    ipRefDescriptors      : ITcVnImage;
    nRefImageWidth        : UDINT;
    nRefImageHeight       : UDINT;
    ipEdgePoints          : Reference To ITcVnContainer;
    fMaxDist              : REAL;
    fMaxKnnRatio          : REAL;
    eNormType             : ETcVnNormalizationType;
    eAlgorithm            : ETcVnEstimationAlgorithm;
    fReprojThreshold      : LREAL;
    nMaxIterations        : UDINT;
    fConfidence           : LREAL;
    ipGoodMatches         : Reference To ITcVnContainer;
    ipInlierMask          : Reference To ITcVnContainer;
    hrPrev                : HRESULT;
END_VAR
VAR_OUTPUT
    aPerspectiveTransform : TcVnMatrix3x3_LREAL;
    nNumberOfGoodMatches  : UDINT;
    nNumberOfInliers      : UDINT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcKeyPoints	ITcVnContainer [▶ 385]	KeyPoints of the source image, e.g. (ContainerType_Vector_TcVnKeyPoint)
ipSrcDescriptors	ITcVnImage [▶ 426]	Descriptors of the source image KeyPoints
ipRefKeyPoints	ITcVnContainer [▶ 385]	KeyPoints of the reference image (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage [▶ 426]	Descriptors of the reference image KeyPoints
nRefImageWidth	UDINT	Width of the reference image in pixels
nRefImageHeight	UDINT	Height of the reference image in pixels
ipEdgePoints	Reference To ITcVnContainer [▶ 385]	Returns a container with the 4 edge points of the reference image transformed into the coordinates of the source image (ContainerType_Vector_TcVnPoint2_REAL)
fMaxDist	REAL	Maximum allowed descriptor distance (-1 disables this filter criterion)
fMaxKnnRatio	REAL	Maximum allowed distance ratio between first and second best match ([0..1], -1 disables this filter criterion)
eNormType	ETcVnNormalizationType [▶ 221]	Normalization type used for descriptor matching (HAMMING recommended for AKAZE, ORB, and BRISK. HAMMING2 is recommended for ORB if the ORB nBriefPoints parameter is 3 or 4.)
eAlgorithm	ETcVnEstimationAlgorithm [▶ 211]	Estimation algorithm used for computing the perspective transformation between the two point sets
fReprojThreshold	LREAL	Maximum allowed reprojection error to treat a point pair as an inlier (only used if eAlgorithm is RANSAC, RHO)
nMaxIterations	UDINT	Maximum number of RANSAC iterations
fConfidence	LREAL	Confidence (0..1)
ipGoodMatches	Reference To ITcVnContainer [▶ 385]	Returns a container with good matches (optional, set to 0 if not required; ContainerType_Vector_TcVnDMatch)
ipInlierMask	Reference To ITcVnContainer [▶ 385]	Returns a mask marking the inliers (optional, set to 0 if not required; ContainerType_Vector_SINT; only for RANSAC, LMEDS)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Outputs

Name	Type	Description
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 173]	Returns the perspective transformation matrix, which transforms the reference points to the source points
nNumberOfGoodMatches	UDINT	Return the number of good matches
nNumberOfInliers	UDINT	Return the number of inlier matches

Return value

HRESULT [▶ 150]

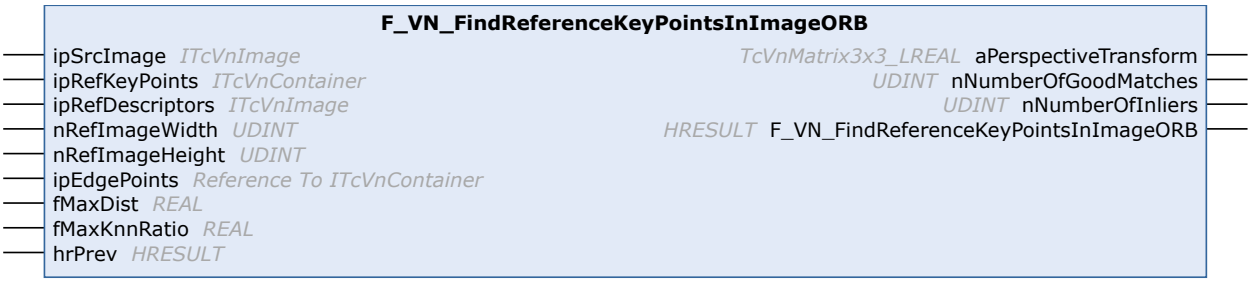
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.8 F_VN_FindReferenceKeyPointsInImageORB



Searches a reference image, represented by its ORB keypoints and corresponding descriptors, in a given source image (uses the default TcVnParamsORB parameters).
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_FindReferenceKeyPointsInImageORB : HRESULT
VAR_INPUT
    ipSrcImage            : ITcVnImage;
    ipRefKeyPoints        : ITcVnContainer;
    ipRefDescriptors      : ITcVnImage;
    nRefImageWidth        : UDINT;
    nRefImageHeight       : UDINT;
    ipEdgePoints          : Reference To ITcVnContainer;
    fMaxDist              : REAL;
    fMaxKnnRatio          : REAL;
    hrPrev                : HRESULT;
END_VAR
VAR_OUTPUT
    aPerspectiveTransform : TcVnMatrix3x3_LREAL;
    nNumberOfGoodMatches  : UDINT;
    nNumberOfInliers      : UDINT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipRefKeyPoints	ITcVnContainer [▶ 385]	KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsORB (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage [▶ 426]	Descriptors of the reference image KeyPoints, e.g. computed with F_VN_KeyPointsAndDescriptorsORB
nRefImageWidth	UDINT	Width of the reference image in pixels
nRefImageHeight	UDINT	Height of the reference image in pixels
ipEdgePoints	Reference To ITcVnContainer [▶ 385]	Returns a container with the 4 edge points of the reference image transformed into ipSrcImage coordinates (ContainerType_Vector_TcVnPoint2_REAL)
fMaxDist	REAL	Maximum allowed descriptor distance (-1 disables this filter criterion)
fMaxKnnRatio	REAL	Maximum allowed distance ratio between first and second best match ([0..1], -1 disables this filter criterion)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Outputs

Name	Type	Description
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 173]	Returns the perspective transformation matrix, which transforms the reference points to the source points
nNumberOfGoodMatches	UDINT	Return the number of good matches
nNumberOfInliers	UDINT	Return the number of inlier matches

Return value

HRESULT [▶ 150]

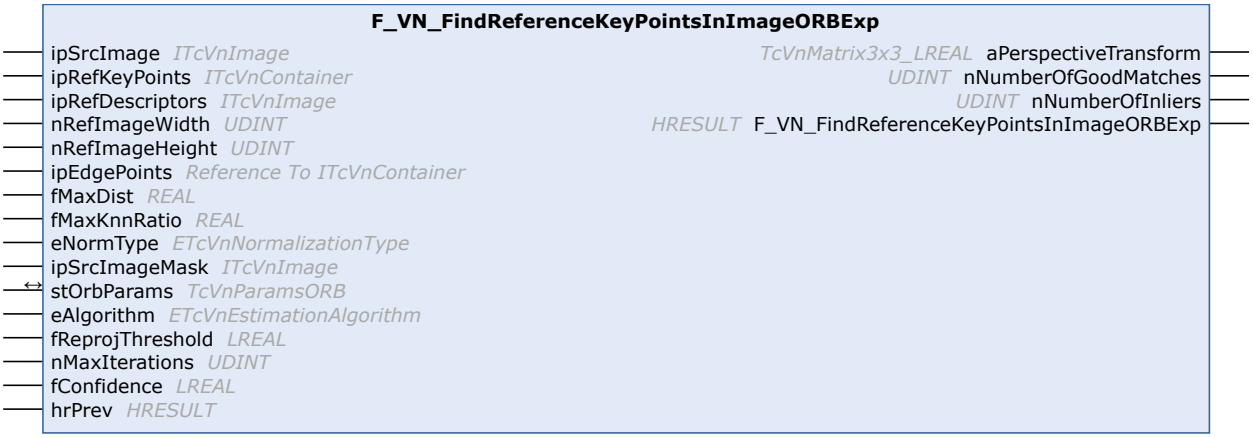
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.9 F_VN_FindReferenceKeyPointsInImageORBExp



Searches a reference image, represented by its ORB keypoints and corresponding descriptors, in a given source image. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_FindReferenceKeyPointsInImageORBExp : HRESULT
VAR_INPUT
    ipSrcImage            : ITcVnImage;
    ipRefKeyPoints        : ITcVnContainer;
    ipRefDescriptors      : ITcVnImage;
    nRefImageWidth        : UDINT;
    nRefImageHeight       : UDINT;
    ipEdgePoints          : Reference To ITcVnContainer;
    fMaxDist              : REAL;
    fMaxKnnRatio          : REAL;
    eNormType             : ETcVnNormalizationType;
    ipSrcImageMask        : ITcVnImage;
END_VAR
VAR_IN_OUT
    stOrbParams           : TcVnParamsORB;
END_VAR
VAR_INPUT
    eAlgorithm            : ETcVnEstimationAlgorithm;
    fReprojThreshold      : LREAL;
    nMaxIterations        : UDINT;
    fConfidence           : LREAL;
    hrPrev                : HRESULT;
END_VAR
VAR_OUTPUT
    aPerspectiveTransform : TcVnMatrix3x3_LREAL;
    nNumberOfGoodMatches  : UDINT;
    nNumberOfInliers      : UDINT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipRefKeyPoints	ITcVnContainer [▶ 385]	KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsORB (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage [▶ 426]	Descriptors of the reference image KeyPoints, e.g. computed with F_VN_KeyPointsAndDescriptorsORB
nRefImageWidth	UDINT	Width of the reference image in pixels
nRefImageHeight	UDINT	Height of the reference image in pixels
ipEdgePoints	Reference To ITcVnContainer [▶ 385]	Returns a container with the 4 edge points of the reference image transformed into ipSrcImage coordinates (ContainerType_Vector_TcVnPoint2_REAL)
fMaxDist	REAL	Maximum allowed descriptor distance (-1 disables this filter criterion)
fMaxKnnRatio	REAL	Maximum allowed distance ratio between first and second best match ([0..1], -1 disables this filter criterion)
eNormType	ETcVnNormalizationType [▶ 221]	Normalization type used for descriptor matching (HAMMING recommended for ORB, HAMMING2 if the ORB nBriefPoints parameter is 3 or 4)
ipSrcImageMask	ITcVnImage [▶ 426]	Mask to specify, where to look for keypoints in ipSrcImage (set to 0 if no mask required)
eAlgorithm	ETcVnEstimationAlgorithm [▶ 211]	Estimation algorithm used for computing the perspective transformation between the two point sets
fReprojThreshold	LREAL	Maximum allowed reprojection error to treat a point pair as an inlier (only used if eAlgorithm is RANSAC, RHO)
nMaxIterations	UDINT	Maximum number of RANSAC iterations
fConfidence	LREAL	Confidence (0..1)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stOrbParams	TcVnParamsORB [▶ 254]	Parameters to configure the keypoint and descriptor computation for ipSrcImage (resulting descriptors must be compatible to ipRefDescriptors!)

Outputs

Name	Type	Description
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 173]	Returns the perspective transformation matrix, which transforms the reference points to the source points
nNumberOfGoodMatches	UDINT	Return the number of good matches
nNumberOfInliers	UDINT	Return the number of inlier matches

Return value

HRESULT [▶ 150]

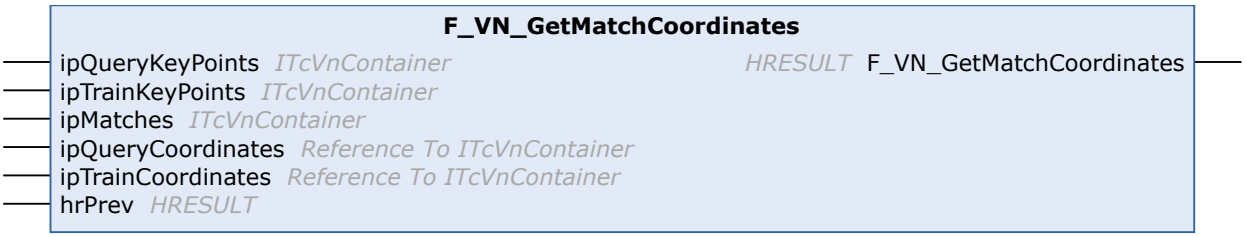
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.10 F_VN_GetMatchCoordinates



Return the coordinates of keypoints that match each other.

Syntax

Definition:

```
FUNCTION F_VN_GetMatchCoordinates : HRESULT
VAR_INPUT
    ipQueryKeyPoints    : ITcVnContainer;
    ipTrainKeyPoints    : ITcVnContainer;
    ipMatches           : ITcVnContainer;
    ipQueryCoordinates  : Reference To ITcVnContainer;
    ipTrainCoordinates  : Reference To ITcVnContainer;
    hrPrev              : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipQueryKeyPoints	ITcVnContainer [▶ 385]	Container with query Keypoints (i.e. keypoints of a reference/ template image) of type CTcVnContainer_Vector_TcVnKeyPoint
ipTrainKeyPoints	ITcVnContainer [▶ 385]	Container with training Keypoints (i.e. keypoints of a source/ input image) of type CTcVnContainer_Vector_TcVnKeyPoint
ipMatches	ITcVnContainer [▶ 385]	Container with matches between query and train Keypoints (CTcVnContainer_Vector_TcVnDMatch).
ipQueryCoordinates	Reference To ITcVnContainer [▶ 385]	Returns a container with the coordinates of keypoints from the query Keypoints that exist in ipMatches (CTcVnContainer_Vector_TcVnPoint2_REAL)
ipTrainCoordinates	Reference To ITcVnContainer [▶ 385]	Returns a container with the coordinates of keypoints from the training Keypoints that exist in ipMatches (CTcVnContainer_Vector_TcVnPoint2_REAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

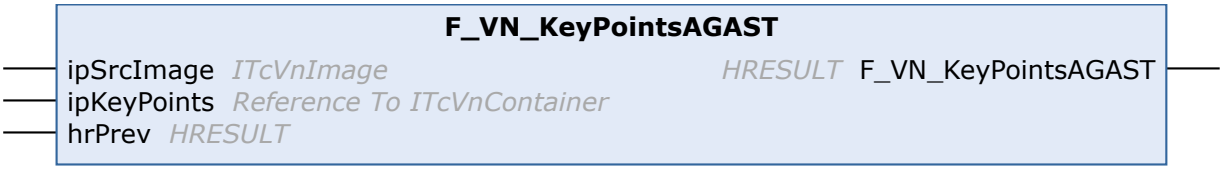
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.11 F_VN_KeyPointsAGAST



Detects keypoints using the AGAST method.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_KeyPointsAGAST : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipKeyPoints : Reference To ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

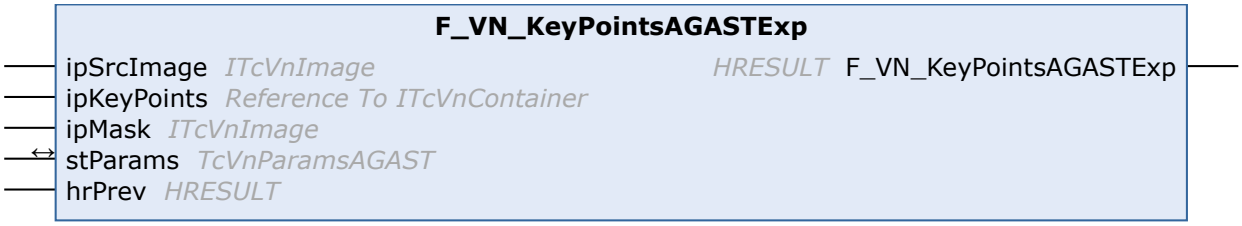
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.12 F_VN_KeyPointsAGASTExp



Detects keypoints using the AGAST method. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_KeyPointsAGASTExp : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipKeyPoints : Reference To ITcVnContainer;
    ipMask      : ITcVnImage;
END_VAR
VAR_IN_OUT
    stParams    : TcVnParamsAGAST;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipMask	ITcVnImage [▶ 426]	Mask to specify, where to look for keypoints (set to 0 if no mask required)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsAGAST [▶ 246]	Additional expert parameters

Return value

HRESULT [▶ 150]

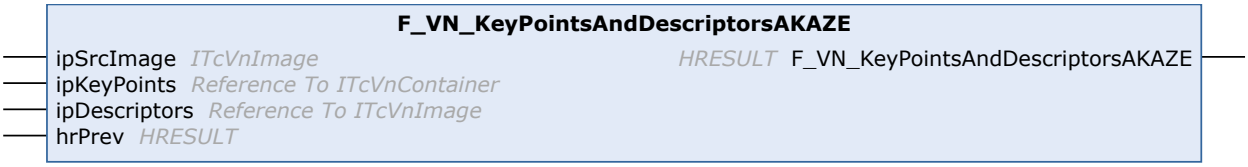
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.13 F_VN_KeyPointsAndDescriptorsAKAZE



Detects keypoints and compute descriptors using the AKAZE method.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_KeyPointsAndDescriptorsAKAZE : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipKeyPoints   : Reference To ITcVnContainer;
    ipDescriptors : Reference To ITcVnImage;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	Reference To ITcVnImage [▶ 426]	Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

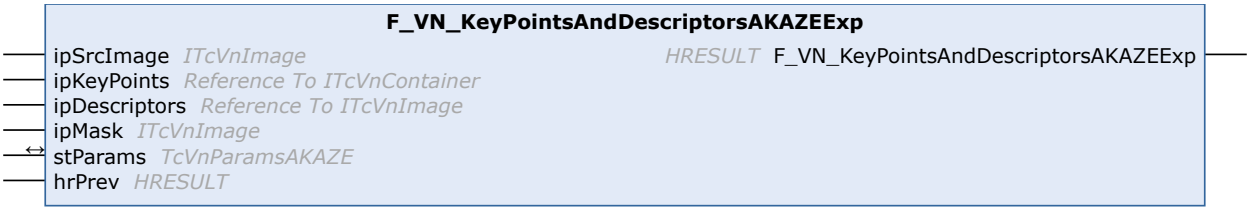
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.14 F_VN_KeyPointsAndDescriptorsAKAZEExp



Detects keypoints and compute descriptors using the AKAZE method. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_KeyPointsAndDescriptorsAKAZEExp : HRESULT
VAR_INPUT
    ipSrcImage     : ITcVnImage;
    ipKeyPoints    : Reference To ITcVnContainer;
    ipDescriptors : Reference To ITcVnImage;
    ipMask         : ITcVnImage;
END_VAR
VAR_IN_OUT
    stParams       : TcVnParamsAKAZE;
END_VAR
VAR_INPUT
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	Reference To ITcVnImage [▶ 426]	Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
ipMask	ITcVnImage [▶ 426]	Mask to specify, where to look for keypoints (set to 0 if no mask required)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsAKAZE [▶ 247]	Additional expert parameters

Return value

HRESULT [▶ 150]

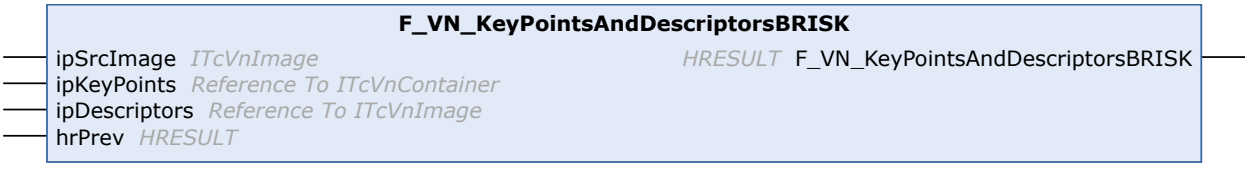
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.15 F_VN_KeyPointsAndDescriptorsBRISK



Detects keypoints and compute descriptors using the BRISK method.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_KeyPointsAndDescriptorsBRISK : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipKeyPoints   : Reference To ITcVnContainer;
    ipDescriptors : Reference To ITcVnImage;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	Reference To ITcVnImage [▶ 426]	Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.16 F_VN_KeyPointsAndDescriptorsBRISKExp

Detects keypoints and compute descriptors using the BRISK method. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_KeyPointsAndDescriptorsBRISKExp : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipKeyPoints   : Reference To ITcVnContainer;
    ipDescriptors : Reference To ITcVnImage;
    ipMask        : ITcVnImage;
END_VAR
VAR_IN_OUT
    stParams      : TcVnParamsBRISK;
```

```
END_VAR
VAR_INPUT
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	Reference To ITcVnImage [▶ 426]	Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
ipMask	ITcVnImage [▶ 426]	Mask to specify, where to look for keypoints (set to 0 if no mask required)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsBRISK [▶ 251]	Additional expert parameters

Return value

HRESULT [▶ 150]

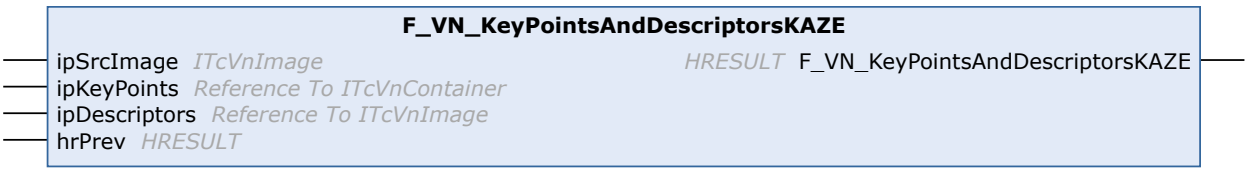
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.17 F_VN_KeyPointsAndDescriptorsKAZE



Detects keypoints and compute descriptors using the KAZE method.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_KeyPointsAndDescriptorsKAZE : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipKeyPoints   : Reference To ITcVnContainer;
    ipDescriptors : Reference To ITcVnImage;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	Reference To ITcVnImage [▶ 426]	Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

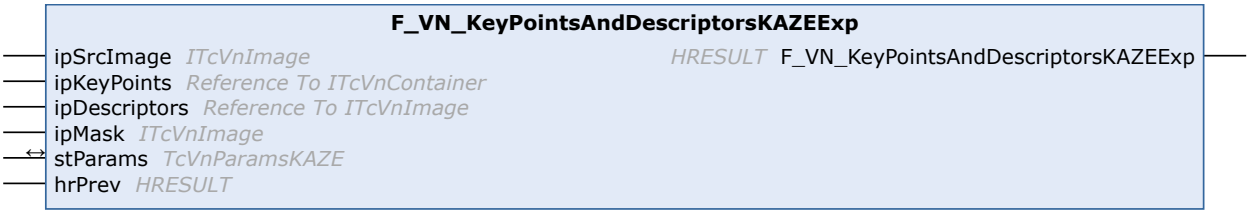
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.18 F_VN_KeyPointsAndDescriptorsKAZEExp



Detects keypoints and compute descriptors using the KAZE method. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_KeyPointsAndDescriptorsKAZEExp : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipKeyPoints   : Reference To ITcVnContainer;
    ipDescriptors : Reference To ITcVnImage;
    ipMask        : ITcVnImage;
END_VAR
VAR_IN_OUT
    stParams      : TcVnParamsKAZE;
END_VAR
VAR_INPUT
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	Reference To ITcVnImage [▶ 426]	Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
ipMask	ITcVnImage [▶ 426]	Mask to specify, where to look for keypoints (set to 0 if no mask required)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsKAZE [▶ 253]	Additional expert parameters

Return value

HRESULT [▶ 150]

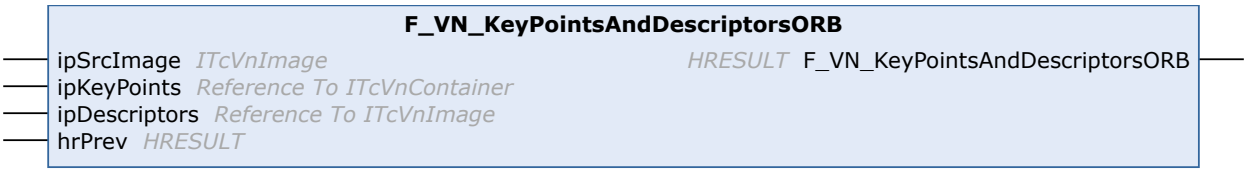
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.19 F_VN_KeyPointsAndDescriptorsORB



Detects keypoints and compute descriptors using the ORB method.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_KeyPointsAndDescriptorsORB : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipKeyPoints   : Reference To ITcVnContainer;
    ipDescriptors : Reference To ITcVnImage;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	Reference To ITcVnImage [▶ 426]	Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.20 F_VN_KeyPointsAndDescriptorsORBExp

Detect keypoints and compute descriptors using the ORB method. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_KeyPointsAndDescriptorsORBExp : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
    ipKeyPoints   : Reference To ITcVnContainer;
    ipDescriptors : Reference To ITcVnImage;
    ipMask        : ITcVnImage;
END_VAR
VAR_IN_OUT
    stParams      : TcVnParamsORB;
END_VAR
VAR_INPUT
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	Reference To ITcVnImage [▶ 426]	Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
ipMask	ITcVnImage [▶ 426]	Mask to specify, where to look for keypoints (set to 0 if no mask required)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsORB [▶ 254]	Additional expert parameters

Return value

HRESULT [▶ 150]

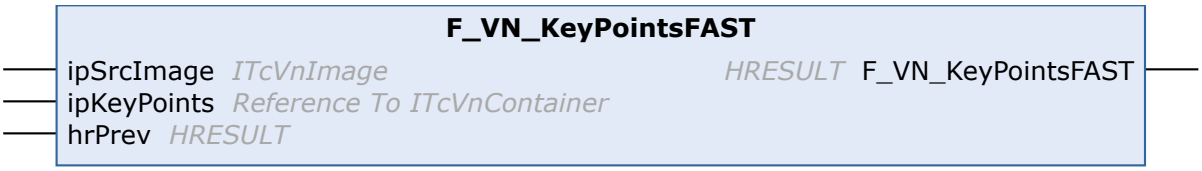
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.21 F_VN_KeyPointsFAST



Detects keypoints using the FAST method.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_KeyPointsFAST : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipKeyPoints : Reference To ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (single-channel).
ipKeyPoints	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

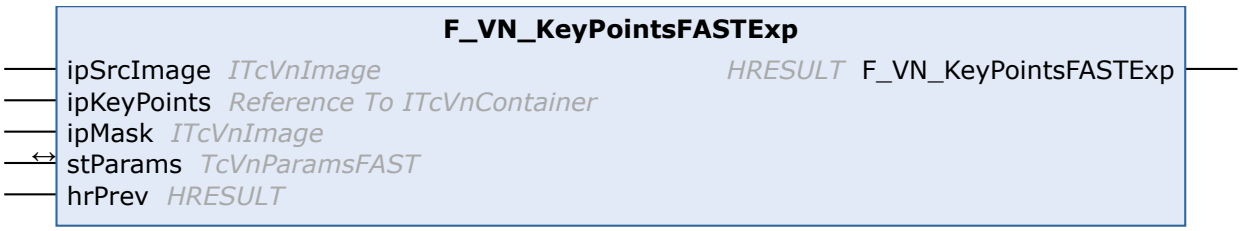
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.22 F_VN_KeyPointsFASTExp



Detects keypoints using the FAST method. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_KeyPointsFASTExp : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipKeyPoints : Reference To ITcVnContainer;
    ipMask      : ITcVnImage;
END_VAR
VAR_IN_OUT
    stParams    : TcVnParamsFAST;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipMask	ITcVnImage [▶ 426]	Mask to specify, where to look for keypoints (set to 0 if no mask required)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsFAST [▶ 252]	Additional expert parameters

Return value

HRESULT [▶ 150]

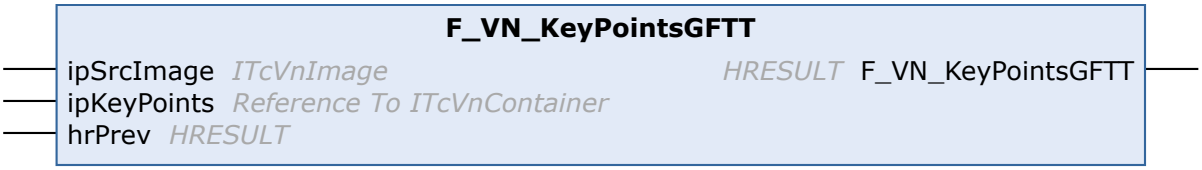
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.23 F_VN_KeyPointsGFTT



Detects keypoints using the GFTT method, which detects strong corners.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_KeyPointsGFTT : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipKeyPoints : Reference To ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

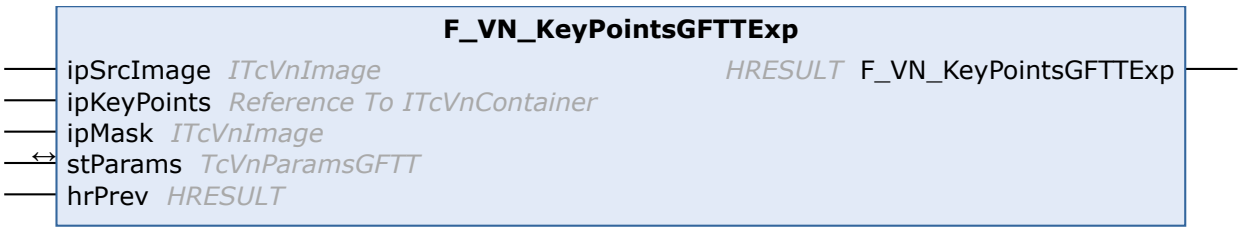
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.24 F_VN_KeyPointsGFTTExp



Detects keypoints using the GFTT method, which detects strong corners. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_KeyPointsGFTTExp : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipKeyPoints : Reference To ITcVnContainer;
    ipMask      : ITcVnImage;
END_VAR
VAR_IN_OUT
    stParams    : TcVnParamsGFTT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipMask	ITcVnImage [▶ 426]	Mask to specify, where to look for keypoints (set to 0 if no mask required)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsGFTT [▶ 252]	Additional expert parameters

Return value

HRESULT [▶ 150]

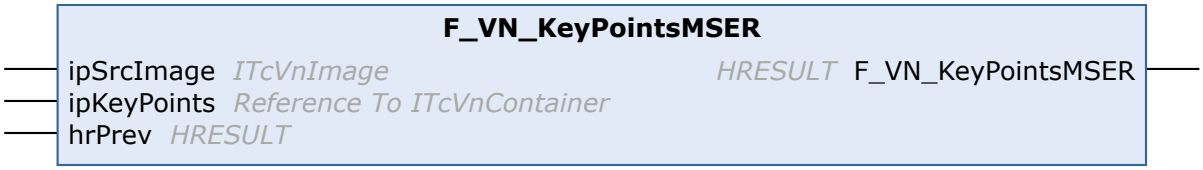
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.25 F_VN_KeyPointsMSER



Detects keypoints using the MSER method.

Syntax

Definition:

```
FUNCTION F_VN_KeyPointsMSER : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipKeyPoints : Reference To ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

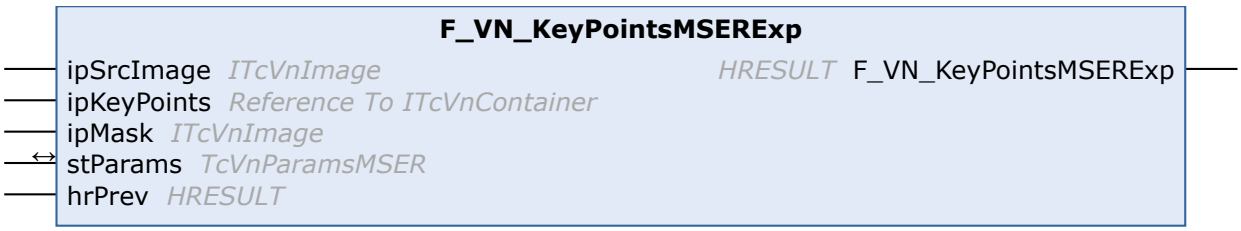
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.26 F_VN_KeyPointsMSERExp



Detects keypoints using the MSER method. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_KeyPointsMSERExp : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipKeyPoints : Reference To ITcVnContainer;
    ipMask      : ITcVnImage;
END_VAR
VAR_IN_OUT
    stParams    : TcVnParamsMSER;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipMask	ITcVnImage [▶ 426]	Mask to specify, where to look for keypoints (set to 0 if no mask required)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsMSER [▶ 253]	Additional expert parameters

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.27 F_VN_KeyPointsSB

Detects keypoints using a Simple Blob method - several iterations apply different thresholds to source image - connected components (blobs) are detected - the center and radius of the blobs are returned as keypoints Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_KeyPointsSB : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipKeyPoints : Reference To ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

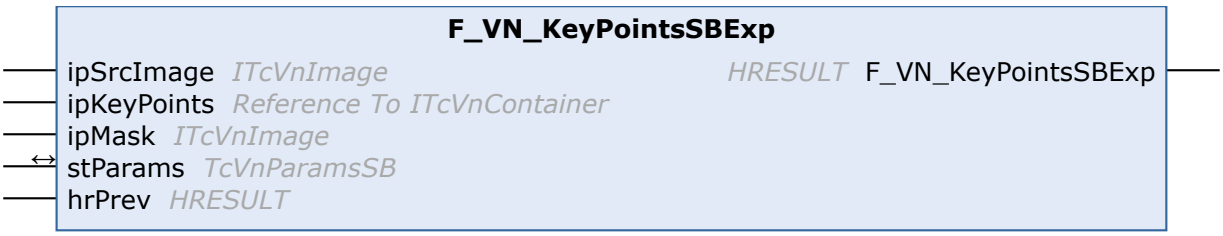
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.28 F_VN_KeyPointsSBExp



Detects keypoints using a Simple Blob method (expert function) - several iterations apply different thresholds to source image - connected components (blobs) are detected - the center and radius of the blobs are returned as keypoints
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_KeyPointsSBExp : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipKeyPoints : Reference To ITcVnContainer;
    ipMask      : ITcVnImage;
END_VAR
VAR_IN_OUT
    stParams    : TcVnParamsSB;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipKeyPoints	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipMask	ITcVnImage [▶ 426]	Mask to specify, where to look for keypoints (optional, set to 0 if not required)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsSB [▶ 255]	Several parameters to filter the detected blobs

Return value

HRESULT [▶ 150]

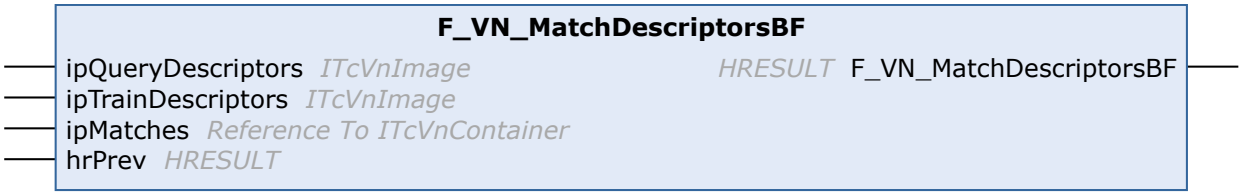
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.29 F_VN_MatchDescriptorsBF



Match descriptors using a brute force approach.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_MatchDescriptorsBF : HRESULT
VAR_INPUT
    ipQueryDescriptors : ITcVnImage;
    ipTrainDescriptors : ITcVnImage;
    ipMatches          : Reference To ITcVnContainer;
    hrPrev             : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipQueryDescriptors	ITcVnImage [▶ 426]	Query descriptors (i.e. descriptors of a reference/template image)
ipTrainDescriptors	ITcVnImage [▶ 426]	Training descriptors (i.e. descriptors of a source/input image)
ipMatches	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the descriptor matches (ContainerType_Vector_TcVnDMatch; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

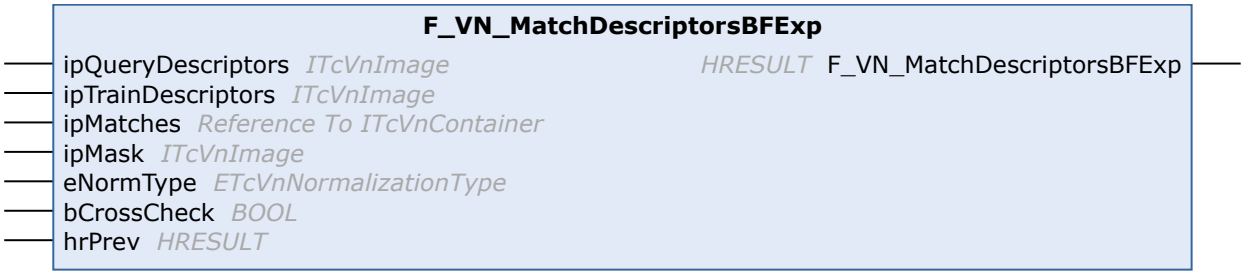
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.30 F_VN_MatchDescriptorsBFExp



Match descriptors using a brute force approach. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_MatchDescriptorsBFExp : HRESULT
VAR_INPUT
    ipQueryDescriptors : ITcVnImage;
    ipTrainDescriptors : ITcVnImage;
    ipMatches          : Reference To ITcVnContainer;
    ipMask             : ITcVnImage;
    eNormType          : ETcVnNormalizationType;
    bCrossCheck        : BOOL;
    hrPrev             : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipQueryDescriptors	ITcVnImage [▶ 426]	Query descriptors (i.e. descriptors of a reference/template image)
ipTrainDescriptors	ITcVnImage [▶ 426]	Training descriptors (i.e. descriptors of a source/input image)
ipMatches	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the descriptor matches (ContainerType_Vector_TcVnDMatch; Non-zero interface pointers are reused.)
ipMask	ITcVnImage [▶ 426]	Mask to specify permissable matches, i.e. query[i] can be matched with train[j] only if mask[i][j] != 0 (set to 0 if no mask required)
eNormType	ETcVnNormalizationType [▶ 221]	Normalization type (only L1, L2, L2SQR, HAMMING, HAMMING2 supported). HAMMING2 should be used for ORB descriptors if the ORB nBriefPoints parameter is 3 or 4)
bCrossCheck	BOOL	If true, only consistent matches are returned, i.e. query->train and train->query detect the same match
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

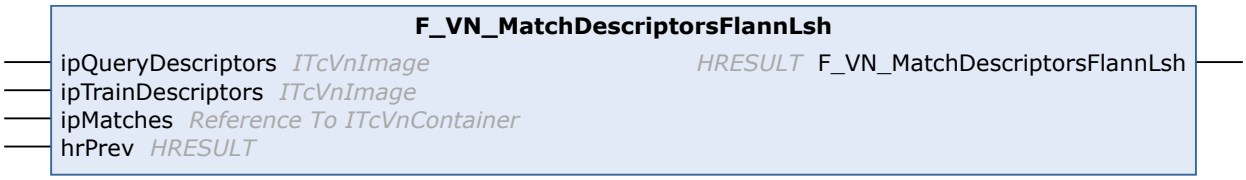
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.31 F_VN_MatchDescriptorsFlannLsh



Match descriptors using a FLANN based approach with LSH index.

Syntax

Definition:

```
FUNCTION F_VN_MatchDescriptorsFlannLsh : HRESULT
VAR_INPUT
    ipQueryDescriptors : ITcVnImage;
    ipTrainDescriptors : ITcVnImage;
    ipMatches          : Reference To ITcVnContainer;
    hrPrev             : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipQueryDescriptors	ITcVnImage [▶ 426]	Query descriptors (i.e. descriptors of a reference/template image)
ipTrainDescriptors	ITcVnImage [▶ 426]	Training descriptors (i.e. descriptors of a source/input image)
ipMatches	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the descriptor matches (ContainerType_Vector_TcVnDMatch; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Anzahl Matches

Die Anzahl der Matches kann bei dieser Funktion verfahrensbedingt von Aufruf zu Aufruf variieren.

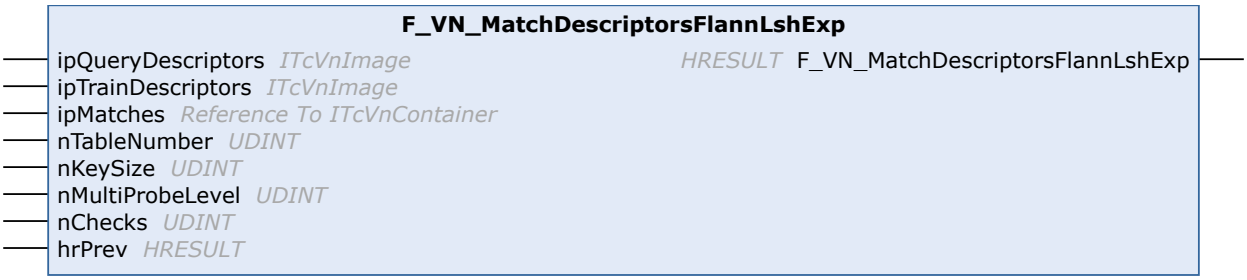
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.32 F_VN_MatchDescriptorsFlannLshExp



Match descriptors using a FLANN based approach with LSH index. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_MatchDescriptorsFlannLshExp : HRESULT
VAR_INPUT
    ipQueryDescriptors : ITcVnImage;
    ipTrainDescriptors : ITcVnImage;
    ipMatches          : Reference To ITcVnContainer;
    nTableNumber       : UDINT;
    nKeySize           : UDINT;
    nMultiProbeLevel   : UDINT;
    nChecks            : UDINT;
    hrPrev             : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipQueryDescriptors	ITcVnImage [▶ 426]	Query descriptors (i.e. descriptors of a reference/template image)
ipTrainDescriptors	ITcVnImage [▶ 426]	Training descriptors (i.e. descriptors of a source/input image)
ipMatches	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the descriptor matches (ContainerType_Vector_TcVnDMatch; Non-zero interface pointers are reused.)
nTableNumber	UDINT	Number of tables
nKeySize	UDINT	Key size
nMultiProbeLevel	UDINT	Multi-probe level
nChecks	UDINT	Maximum number of visited leafs when searching for neighbors
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

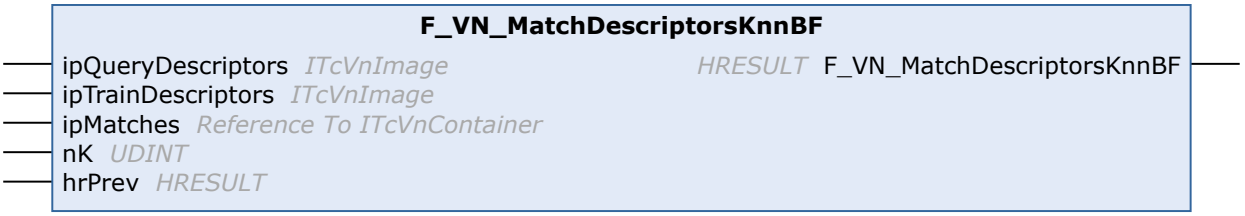
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.33 F_VN_MatchDescriptorsKnnBF



Match descriptors (k nearest neighbors) using a brute force approach.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_MatchDescriptorsKnnBF : HRESULT
VAR_INPUT
    ipQueryDescriptors : ITcVnImage;
    ipTrainDescriptors : ITcVnImage;
    ipMatches          : Reference To ITcVnContainer;
    nK                 : UDINT;
    hrPrev             : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipQueryDescriptors	ITcVnImage [▶ 426]	Query descriptors (i.e. descriptors of a reference/template image)
ipTrainDescriptors	ITcVnImage [▶ 426]	Training descriptors (i.e. descriptors of a source/input image)
ipMatches	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the descriptor matches (ContainerType_Vector_Vector_TcVnDMatch; Non-zero interface pointers are reused.)
nK	UDINT	Number of required best train matches for each query descriptor (i.e. nk := 2)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

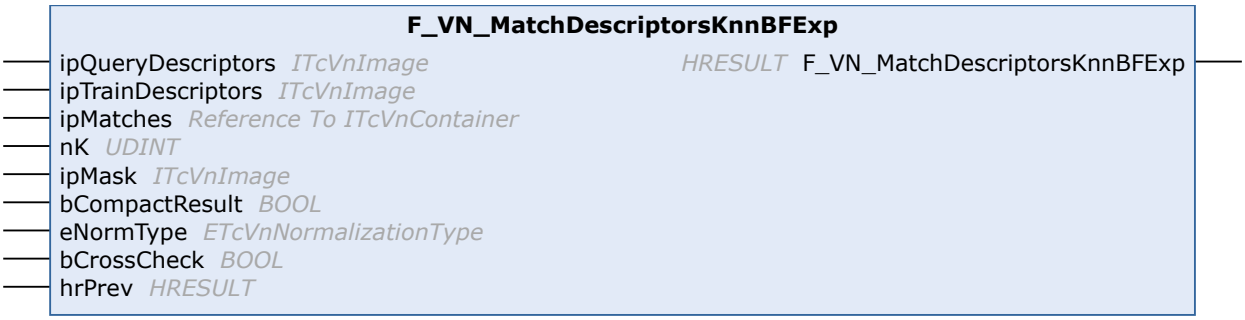
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.34 F_VN_MatchDescriptorsKnnBFExp



Match descriptors (k nearest neighbors) using a brute force approach. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_MatchDescriptorsKnnBFExp : HRESULT
VAR_INPUT
    ipQueryDescriptors : ITcVnImage;
    ipTrainDescriptors : ITcVnImage;
    ipMatches          : Reference To ITcVnContainer;
    nK                 : UDINT;
    ipMask             : ITcVnImage;
    bCompactResult     : BOOL;
    eNormType          : ETcVnNormalizationType;
    bCrossCheck        : BOOL;
    hrPrev             : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipQueryDescriptors	ITcVnImage [▶ 426]	Query descriptors (i.e. descriptors of a reference/template image)
ipTrainDescriptors	ITcVnImage [▶ 426]	Training descriptors (i.e. descriptors of a source/input image)
ipMatches	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the descriptor matches (ContainerType_Vector_Vector_TcVnDMatch; Non-zero interface pointers are reused.)
nK	UDINT	Number of required best train matches for each query descriptor (i.e. nk := 2)
ipMask	ITcVnImage [▶ 426]	Mask to specify permissable matches, i.e. query[i] can be matched with train[j] only if mask[i][j] != 0 (set to 0 if no mask required)
bCompactResult	BOOL	If true, matches vector does not contain entries for fully masked-out query descriptors
eNormType	ETcVnNormalizationType [▶ 221]	Normalization type (only L1, L2, L2SQR, HAMMING, HAMMING2 supported). HAMMING2 should be used for ORB descriptors if the ORB nBriefPoints parameter is 3 or 4)
bCrossCheck	BOOL	If true, only consistent matches are returned, i.e. query->train and train->query detect the same match (only used if nK = 1)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

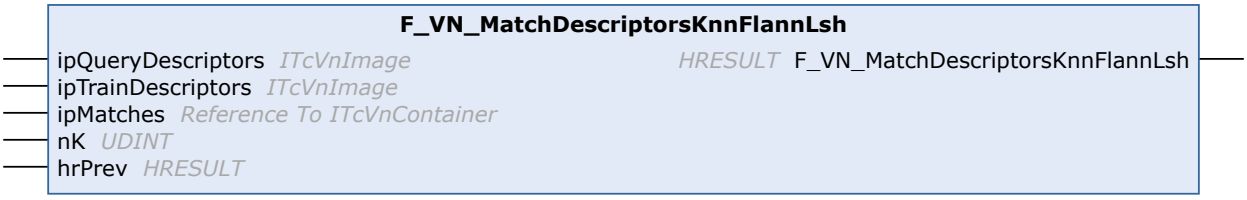
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.35 F_VN_MatchDescriptorsKnnFlannLsh



Match descriptors (k nearest neighbors) using a FLANN based approach with LSH index.

Syntax

Definition:

```
FUNCTION F_VN_MatchDescriptorsKnnFlannLsh : HRESULT
VAR_INPUT
    ipQueryDescriptors : ITcVnImage;
    ipTrainDescriptors : ITcVnImage;
    ipMatches          : Reference To ITcVnContainer;
```

```
    nK                  : UDINT;
    hrPrev              : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipQueryDescriptors	ITcVnImage [▶ 426]	Query descriptors (i.e. descriptors of a reference/template image)
ipTrainDescriptors	ITcVnImage [▶ 426]	Training descriptors (i.e. descriptors of a source/input image)
ipMatches	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the descriptor matches (ContainerType_Vector_Vector_TcVnDMatch; Non-zero interface pointers are reused.)
nK	UDINT	Number of required best train matches for each query descriptor (i.e. nk := 2)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

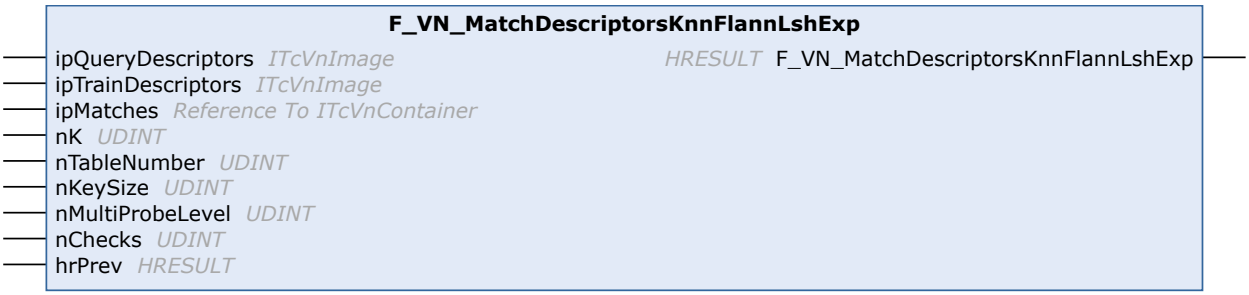
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.36 F_VN_MatchDescriptorsKnnFlannLshExp



Match descriptors (k nearest neighbors) using a FLANN based approach with LSH index. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_MatchDescriptorsKnnFlannLshExp : HRESULT
VAR_INPUT
    ipQueryDescriptors : ITcVnImage;
    ipTrainDescriptors : ITcVnImage;
    ipMatches          : Reference To ITcVnContainer;
    nK                 : UDINT;
    nTableNumber       : UDINT;
    nKeySize           : UDINT;
    nMultiProbeLevel   : UDINT;
    nChecks            : UDINT;
    hrPrev             : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipQueryDescriptors	ITcVnImage [▶ 426]	Query descriptors (i.e. descriptors of a reference/template image)
ipTrainDescriptors	ITcVnImage [▶ 426]	Training descriptors (i.e. descriptors of a source/input image)
ipMatches	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the descriptor matches (ContainerType_Vector_Vector_TcVnDMatch; Non-zero interface pointers are reused.)
nK	UDINT	Number of required best train matches for each query descriptor (i.e. nk := 2)
nTableNumber	UDINT	Number of tables
nKeySize	UDINT	Key size
nMultiProbeLevel	UDINT	Multi-probe level
nChecks	UDINT	Maximum number of visited leafs when searching for neighbors
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

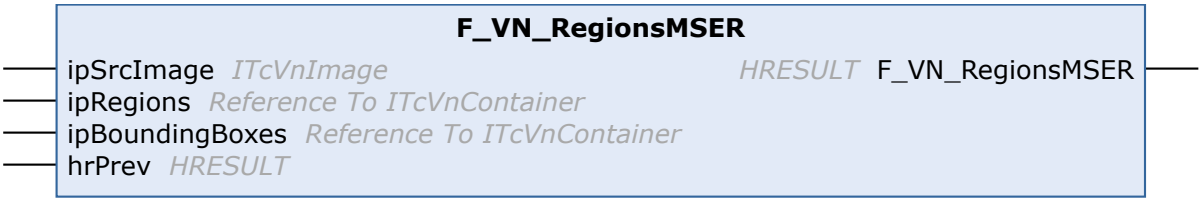
Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.37 F_VN_RegionsMSER



Detects regions using the MSER method.

Syntax

Definition:

```
FUNCTION F_VN_RegionsMSER : HRESULT
VAR_INPUT
    ipSrcImage      : ITcVnImage;
    ipRegions       : Reference To ITcVnContainer;
    ipBoundingBoxes : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipRegions	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the region points (ContainerType_Vector_Vector_TcVnPoint2_DINT; Non-zero interface pointers are reused.)
ipBoundingBoxes	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the region bounding boxes (ContainerType_Vector_TcVnRectangle_DINT; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.18.38 F_VN_RegionsMSERExp

Detects regions using the MSER method. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_RegionsMSERExp : HRESULT
VAR_INPUT
    ipSrcImage      : ITcVnImage;
    ipRegions       : Reference To ITcVnContainer;
    ipBoundingBoxes : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    stParams        : TcVnParamsMSER;
END_VAR
VAR_INPUT
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipRegions	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the region points (ContainerType_Vector_Vector_TcVnPoint2_DINT; Non-zero interface pointers are reused.)
ipBoundingBoxes	Reference To ITcVnContainer [▶ 385]	Returns a container which is filled with the region bounding boxes (ContainerType_Vector_TcVnRectangle_DINT; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stParams	TcVnParamsMSER [▶ 253]	Additional expert parameters

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19 Machine Learning

Diese Gruppe enthält Funktionen zum Erstellen, Trainieren und Anwenden von Machine Learning Modellen sowie Funktionen zur Merkmalsvorverarbeitung. Eine kurze Übersicht der Anwendungsmöglichkeiten von Machine Learning [▶ 153] und weitere Beschreibungen finden Sie unter dem Softwarekonzept [▶ 146].

Funktionen zum Erstellen eines Modells:

- F_VN_CreateBoostClassifier(Exp) [▶ 1414]
- F_VN_CreateKmppModel(Exp) [▶ 1419]
- F_VN_CreateKnnModel [▶ 1423]
- F_VN_CreateLbgModel(Exp) [▶ 1424]
- F_VN_CreateLdaTransform [▶ 1428]
- F_VN_CreateLdaTransformViaComponentNum [▶ 1430]
- F_VN_CreateNbcModel [▶ 1432]
- F_VN_CreatePcaTransform [▶ 1433]
- F_VN_CreatePcaTransformViaComponentNum [▶ 1435]
- F_VN_CreatePcaTransformViaVariance [▶ 1436]
- F_VN_CreateRTreesModel(Exp) [▶ 1438]
- F_VN_CreateStaModel(Exp) [▶ 1443]

- F_VN_CreateSvmModel(Exp) [▶ 1449]
- F_VN_CreateSvmSgdClassifier(Exp) [▶ 1458]

Funktionen zum Trainieren eines Modells:

- F_VN_TrainBatch [▶ 1500]
- F_VN_TrainBatchClusters [▶ 1502]
- F_VN_TrainSample [▶ 1504]
- F_VN_TrainSampleClass [▶ 1505]
- F_VN_TrainSampleCluster [▶ 1507]
- F_VN_TrainSampleScalar [▶ 1508]
- F_VN_TrainSampleVector [▶ 1510]

Funktionen zum Erhalt von Informationen zu den Cluster Modellen:

- F_VN_GetClusterCenter [▶ 1470]
- F_VN_GetClusterNum [▶ 1472]

Funktionen zum Anwenden von Modellen auf einem Sample:

- F_VN_GetSampleCluster(Exp) [▶ 1475]
- F_VN_GetSampleNovelty [▶ 1478]
- F_VN_PredictSampleClass (Exp) [▶ 1491]
- F_VN_PredictSampleScalar(Exp) [▶ 1494]
- F_VN_PredictSampleVector(Exp) [▶ 1497]

Funktionen zum Anwenden von Modellen auf einem Batch:

- F_VN_GetBatchClusters(Exp) [▶ 1466]
- F_VN_GetBatchNovelties [▶ 1469]
- F_VN_PredictBatch(Exp) [▶ 1487]

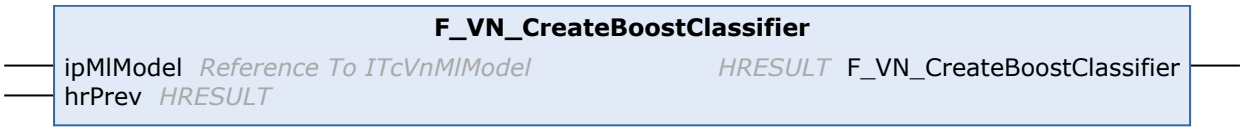
Funktionen zum Anwenden von Modellen zur Merkmalstransformation / Dimensionsreduktion:

- F_VN_FeatureTransform [▶ 1465]
- F_VN_InverseFeatureTransform [▶ 1486]

Funktionen zur Merkmalsnormalisierung:

- F_VN_GetFeatureScales [▶ 1473]
- F_VN_FeatureScaling(Exp) [▶ 1461]
- F_VN_InverseFeatureScaling(Exp) [▶ 1479]
- F_VN_InverseFeatureScaling(Exp)_REAL [▶ 1481]

6.1.4.19.1 F_VN_CreateBoostClassifier



Create a Boost classifier using the default parameters. The initial reference count is set to one if a new model is created and kept, otherwise. The Boost classifier is only applicable to binary classification problems. It learns to distinguish between samples labelled with two user-defined class labels by incrementally adding weak classifiers to improve the classification results. Models of this type neither support on-line training (sample by sample) nor retraining.

Syntax

Definition:

```
FUNCTION F_VN_CreateBoostClassifier : HRESULT
VAR_INPUT
    ipMlModel : Reference To ITcVnMlModel;
    hrPrev    : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created model (Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreateBoostClassifier` erstellt ein Boost-Klassifikator Modell.

Boost-Klassifikator Modelle

Das Boost-Klassifikator Modell ist nur für binäre Klassifizierungsprobleme geeignet. Dieses Modell lernt, zwischen Samples zu unterscheiden, die mit zwei benutzerdefinierten Klassenlabels gekennzeichnet sind, indem es schrittweise schwache Klassifikatoren hinzufügt, um die Klassifizierungsergebnisse zu verbessern.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Expert-Parameter

Die Expert-Varianten F_VN_CreateBoostClassifierExp [▶ 1416] und F_VN_CreateBoostClassifierExp2 [▶ 1417] enthalten zusätzliche Parameter.

Anwendung

Ein Boost-Klassifikator Modell kann z.B. so erstellt werden:

```
hr := F_VN_CreateBoostClassifier(
    ipMlModel   := ipMlModel,
    hrPrev      := hr);
```

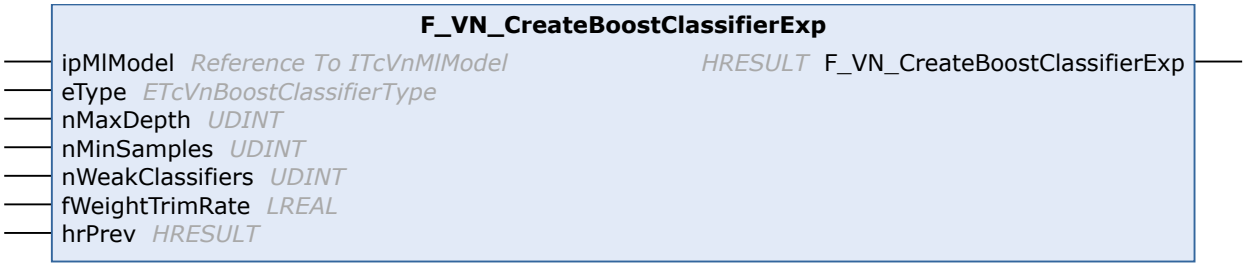
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.2 F_VN_CreateBoostClassifierExp



Create a Boost classifier. The initial reference count is set to one if a new model is created and kept, otherwise. The Boost classifier is only applicable to binary classification problems. It learns to distinguish between samples labelled with two user-defined class labels by incrementally adding weak classifiers to improve the classification results. Models of this type neither support on-line training (sample by sample) nor retraining. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_CreateBoostClassifierExp : HRESULT
VAR_INPUT
    ipMlModel        : Reference To ITcVnMlModel;
    eType            : ETcVnBoostClassifierType;
    nMaxDepth        : UDINT;
    nMinSamples      : UDINT;
    nWeakClassifiers : UDINT;
    fWeightTrimRate  : LREAL;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created model (Non-zero interface pointers are reused.)
eType	ETcVnBoostClassifierType [▶ 177]	Learning algorithm type (default: TCVN_BCT_REAL)
nMaxDepth	UDINT	Maximum tree depth (default: 1)
nMinSamples	UDINT	Minimum number of samples within a node required for splitting (default: 10)
nWeakClassifiers	UDINT	Number of weak classifiers (default: 100)
fWeightTrimRate	LREAL	Weight threshold used during training (off: 0; default: 0.95).
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreateBoostClassifierExp` ist eine Expert-Variante von F_VN_CreateBoostClassifier [▶ 1414]. Sie enthält zusätzliche Parameter.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Lernverfahren

Mittels `eType` kann der Lernalgorithmus ausgewählt werden. Unterstützt werden: Discrete AdaBoost, Real AdaBoost, LogitBoost und Gentle AdaBoost.

Maximale Baumtiefe

`nMaxDepth` gibt an, wie viele Entscheidungsebenen es maximal in einem Baum gibt. Üblicherweise werden für Boost-Klassifikatoren Entscheidungsbäume mit der Baumtiefe 1 verwendet. Höhere Werte sind aber möglich.

Minimale Sample-Anzahl innerhalb eines Knotens

`nMinSamples` ist die minimale Anzahl an Samples, die während der Baumbildung in einem Knoten verbleiben muss. Aufspaltungen, die diese Anzahl unterschreiten würden, werden nicht durchgeführt.

Anzahl an Entscheidungsbäumen

`nWeakClassifiers` gibt die Anzahl an Entscheidungsbäumen (schwache Lerner) an, die für den Klassifikator durch Boosting kombiniert werden.

Laufzeitoptimierung

Mit `fWeightTrimRate` kann die Anzahl an Samples, die für die einzelnen Lernschritte genutzt werden, anhand ihrer automatisch ermittelten Wichtungen beschränkt werden. Zulässig sind Werte im Intervall von 0 bis 1, wobei nur Samples mit Wichtungen > 1 - `fWeightTrimRate` im folgenden Lernschritt berücksichtigt werden. Der Wert 0 schaltet diese Optimierung ab.

Anwendung

Ein Boost-Klassifikator Modell kann z.B. so erstellt werden:

```
hr := F_VN_CreateBoostClassifierExp(
    ipMlModel           := ipMlModel,
    eType               := TCVN_BCT_REAL,
    nMaxDepth           := 1,
    nMinSamples         := 10,
    nWeakClassifiers    := 100,
    fWeightTrimRate     := 0.95,
    hrPrev              := hr);
```

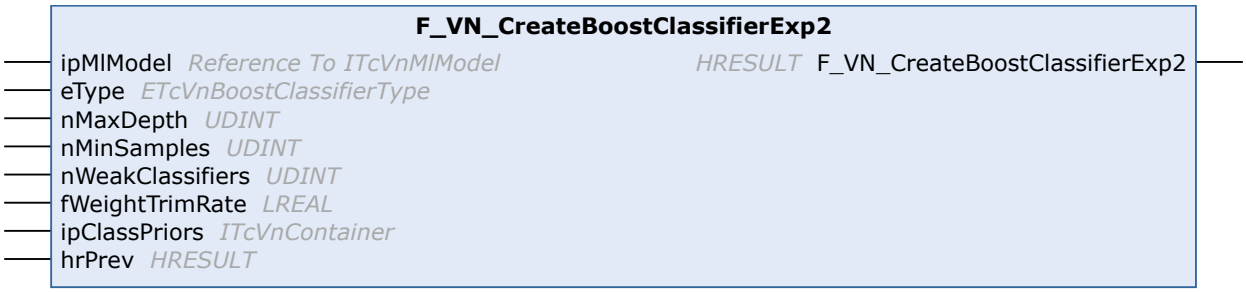
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.3 F_VN_CreateBoostClassifierExp2



Create a Boost classifier. The initial reference count is set to one if a new model is created and kept, otherwise. The Boost classifier is only applicable to binary classification problems. It learns to distinguish between samples labelled with two user-defined class labels by incrementally adding weak classifiers to improve the classification results. Models of this type neither support on-line training (sample by sample) nor retraining. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_CreateBoostClassifierExp2 : HRESULT
VAR_INPUT
    ipMlModel        : Reference To ITcVnMlModel;
    eType            : ETcVnBoostClassifierType;
    nMaxDepth        : UDINT;
    nMinSamples      : UDINT;
    nWeakClassifiers : UDINT;
    fWeightTrimRate  : LREAL;
    ipClassPriors    : ITcVnContainer;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created model (Non-zero interface pointers are reused.)
eType	ETcVnBoostClassifierType [▶ 177]	Learning algorithm type (default: TCVN_BCT_REAL)
nMaxDepth	UDINT	Maximum tree depth (default: 1)
nMinSamples	UDINT	Minimum number of samples within a node required for splitting (default: 10)
nWeakClassifiers	UDINT	Number of weak classifiers (default: 100)
fWeightTrimRate	LREAL	Weight threshold used during training (off: 0; default: 0.95).
ipClassPriors	ITcVnContainer [▶ 385]	Class priors (ContainerType_Vector_REAL or ContainerType_Vector_LREAL; only for classifiers; optional, set to 0 if not required or not allowed; default: 0)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreateBoostClassifierExp2` ist eine Expert-Variante von F_VN_CreateBoostClassifier [▶ 1414]. Sie enthält zusätzliche Parameter.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Lernverfahren

Mittels `eType` kann der Lernalgorithmus ausgewählt werden. Unterstützt werden: Discrete AdaBoost, Real AdaBoost, LogitBoost und Gentle AdaBoost.

Maximale Baumtiefe

`nMaxDepth` gibt an, wie viele Entscheidungsebenen es maximal in einem Baum gibt. Üblicherweise werden für Boost-Klassifikatoren Entscheidungsbäume mit der Baumtiefe 1 verwendet. Höhere Werte sind aber möglich.

Minimale Sample-Anzahl innerhalb eines Knotens

`nMinSamples` ist die minimale Anzahl an Samples, die während der Baumbildung in einem Knoten verbleiben muss. Aufspaltungen, die diese Anzahl unterschreiten würden, werden nicht durchgeführt.

Anzahl an Entscheidungsbäumen

`nWeakClassifiers` gibt die Anzahl an Entscheidungsbäumen (schwache Lerner) an, die für den Klassifikator durch Boosting kombiniert werden.

Laufzeitoptimierung

Mit `fWeightTrimRate` kann die Anzahl an Samples, die für die einzelnen Lernschritte genutzt werden, anhand ihrer automatisch ermittelten Wichtungen beschränkt werden. Zulässig sind Werte im Intervall von 0 bis 1, wobei nur Samples mit Wichtungen > 1 - `fWeightTrimRate` im folgenden Lernschritt berücksichtigt werden. Der Wert 0 schaltet diese Optimierung ab.

Class-Priors

`ipClassPriors` ist ein Container, mit dem die Apriori-Wahrscheinlichkeiten der einzelnen Klassen angegeben werden können.

Anwendung

Ein Boost-Klassifikator Modell kann z.B. so erstellt werden:

```
hr := F_VN_CreateBoostClassifierExp2(
    ipMlModel           := ipMlModel,
    eType               := TCVN_BCT_REAL,
    nMaxDepth           := 1,
    nMinSamples         := 10,
    nWeakClassifiers    := 100,
    fWeightTrimRate     := 0.95,
    ipClassPriors       := 0,
    hrPrev              := hr);
```

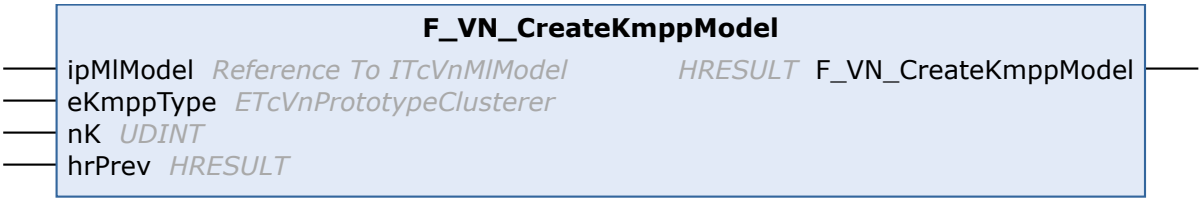
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.4 F_VN_CreateKmppModel



Create a k-means++ model of the specified type. The initial reference count is set to one if a new model is created and kept, otherwise. Models of this type neither support on-line training (sample by sample) nor retraining.

Syntax

Definition:

```
FUNCTION F_VN_CreateKmppModel : HRESULT
VAR_INPUT
    ipMlModel : Reference To ITcVnMlModel;
    eKmppType : ETcVnPrototypeClusterer;
    nK        : UDINT;
    hrPrev    : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created model (Non-zero interface pointers are reused.)
eKmppType	ETcVnPrototypeClusterer [▶ 226]	k-means++ model type
nK	UDINT	Parameter k (number of clusters)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreateKmppModel` erstellt ein K-Means++ (KMPP) Model.

K-Means++ Modelle

K-Means++ kann zum Clustering oder zur Anomalie-Detektion genutzt werden. Es findet Cluster von Datenpunkten. Für das Training werden alle Samples gleichzeitig benötigt (Batch-Training) und ein Nachtrainieren ist nicht möglich.

Die Zuweisung der Cluster-Bezeichnungen (als `DINT`s) kann sich von Training zu Training unterscheiden. Wenn diese Zuweisung deterministisch geschehen soll, muss vorher ein fester Random-Seed mittels F_VN_SetRngSeed [▶ 1588] gesetzt werden.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Modell-Typ

`eKmppType` legt fest, ob K-Means++ zum Clustering (`TCVN_PC_CLUSTERER`) oder zur Anomalie-Detektion (`TCVN_PC_NOVELTY_DETECTOR`) genutzt wird.

Anzahl von Clustern

Die Anzahl der Cluster im K-Means++ Modell wird durch `nK` angegeben und kann nach der Erstellung nicht mehr geändert werden. Beim Training muss die Anzahl der Samples größer-gleich der Anzahl von Clustern sein.

Expert-Parameter

Die Expert-Variante F_VN_CreateKmppModelExp [▶ 1421] enthält zusätzliche Parameter.

Anwendung

Ein K-Means++ Modell für Clustering mit 5 Clustern kann z.B. so erstellt werden:

```
hr := F_VN_CreateKmppModel(
    ipMlModel    := ipMlModel,
    eKmppType    := TCVN_PC_CLUSTERER,
    nK           := 5,
    hrPrev       := hr);
```

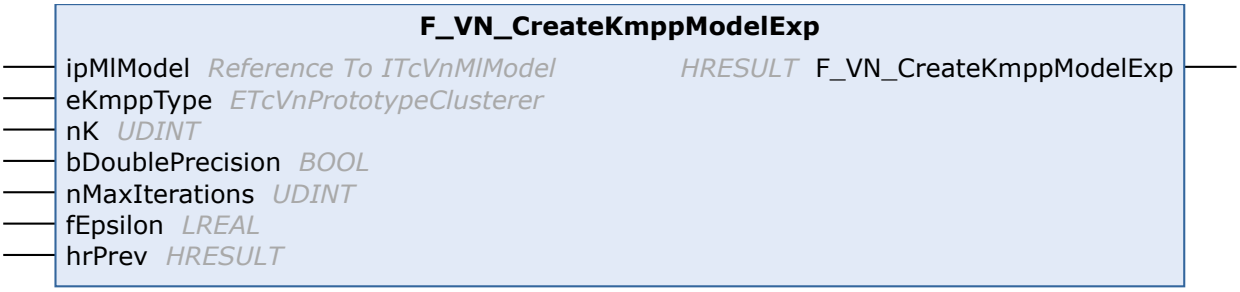
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.5 F_VN_CreateKmppModelExp



Create a k-means++ model of the specified type. The initial reference count is set to one if a new model is created and kept, otherwise. Models of this type neither support on-line training (sample by sample) nor retraining. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_CreateKmppModelExp : HRESULT
VAR_INPUT
    ipMlModel        : Reference To ITcVnMlModel;
    eKmppType        : ETcVnPrototypeClusterer;
    nK               : UDINT;
    bDoublePrecision : BOOL;
    nMaxIterations   : UDINT;
    fEpsilon         : LREAL;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created model (Non-zero interface pointers are reused.)
eKmppType	ETcVnPrototypeClusterer [▶ 226]	k-means++ model type
nK	UDINT	Parameter k (number of clusters)
bDoublePrecision	BOOL	If true, the model is generated with double precision (LREAL) instead of single precision (REAL). (default: FALSE)
nMaxIterations	UDINT	Maximum number of iterations (triggers the usage of the default value of 10 if it equals 0)
fEpsilon	LREAL	Maximum allowed difference of the error between two successive iterations (triggers the usage of the default value of 0.001 if it equals 0)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreateKmppModelExp` ist die Expert-Variante von F_VN_CreateKmppModel [▶ 1419]. Sie enthält zusätzliche Parameter.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Modell-Typ

`eKmppType` legt fest, ob K-Means++ zum Clustering (`TCVN_PC_CLUSTERER`) oder zur Anomalie-Detektion (`TCVN_PC_NOVELTY_DETECTOR`) genutzt wird.

Anzahl von Clustern

Die Anzahl der Cluster im K-Means++ Modell wird durch `nK` angegeben und kann nach der Erstellung nicht mehr geändert werden. Beim Training muss die Anzahl der Samples größer-gleich der Anzahl von Clustern sein.

Höhere Genauigkeit

Wenn `bDoublePrecision` `TRUE` ist, wird `LREAL` als Datentyp für die internen Berechnungen des Modells genutzt; wenn es `FALSE` ist, wird `REAL` genutzt.

Maximale Iterationen

Für die Optimierung werden maximal so viele Iterationen genutzt, wie in `nMaxIterations` angegeben sind. Wenn der Wert `0` ist, wird der jeweilige Default-Wert verwendet.

Abbruchgrenze

Die Optimierung wird abgebrochen, sobald sich der Fehler zwischen zwei Iterationen nicht stärker verändert, als in `fEpsilon` angegeben ist. Wenn der Wert `0` ist, wird der jeweilige Default-Wert verwendet.

Anwendung

Ein K-Means++ Modell für Clustering mit 5 Clustern kann z.B. so erstellt werden:

```
hr := F_VN_CreateKmppModelExp(
    ipMlModel           := ipMlModel,
    eKmppType           := TCVN_PC_CLUSTERER,
    nK                  := 5,
    bDoublePrecision    := FALSE,
    nMaxIterations      := 0,
    fEpsilon            := 0,
    hrPrev              := hr);
```

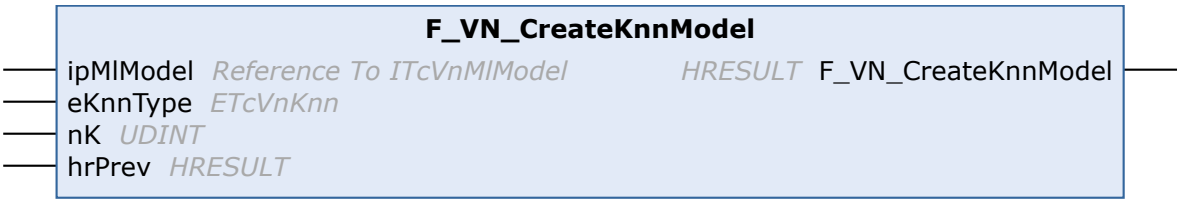
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.6 F_VN_CreateKnnModel



Create a k-nearest neighbors model of the specified type. The initial reference count is set to one if a new model is created and kept, otherwise. Models of this type support on-line training (sample by sample) and retraining. Predictions can only be scalar.

Syntax

Definition:

```
FUNCTION F_VN_CreateKnnModel : HRESULT
VAR_INPUT
    ipMlModel : Reference To ITcVnMlModel;
    eKnnType  : ETcVnKnn;
    nK        : UDINT;
    hrPrev    : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created model (Non-zero interface pointers are reused.)
eKnnType	ETcVnKnn [▶ 219]	k-nearest neighbors model type
nK	UDINT	Parameter k used for prediction (number of considered neighbors)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreateKnnModel` erstellt ein K-Nearest-Neighbor (KNN) Modell.

K-Nearest-Neighbor-Modelle

Beim Training von K-Nearest-Neighbor-Modellen werden alle Trainingssamples 1:1 im Modell gespeichert. Zum Verständnis hilft die Vorstellung der Datenpunkte innerhalb eines mehrdimensionalen Raums (entsprechend der Dimensionalität der Samples). Für die Prädiktion wird eine Anzahl `nK` von Datenpunkten genutzt, die dem Eingangssample am nächsten sind (euklidische Distanz). Diese Datenpunkte werden „nächste Nachbarn“ genannt. Alle anderen Datenpunkte werden bei der Prädiktion ignoriert. Das Prädiktionsergebnis wird aus allen nächsten Nachbarn berechnet, wobei diese unabhängig von der tatsächlichen Distanz zum Eingangssample gleich gewichtet sind.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Modell-Typ

`eKnnType` gibt an, ob das Modell als Klassifikator (`TCVN_KNN_CLASSIFIER`), als Anomalie-Detektion (`TCVN_KNN_NOVELTY_DETECTOR`) oder als Regressor (`TCVN_KNN_REGRESSOR`) genutzt wird.

Anzahl genutzter Nachbarn

`nK` gibt an, wie viele Datenpunkte bei der Prädiktion als nächste Nachbarn genutzt werden. Für die Anomalie-Detektion ist `nK` irrelevant, weil dabei nur der eine nächste Nachbar benötigt wird. Daher sollte in diesem Fall der Wert 1 verwendet werden.

Anwendung

Ein K-Nearest-Neighbor-Modell für die Klassifikation mittels 5 nächster Nachbarn wird z.B. so erstellt:

```
hr := F_VN_CreateKnnModel(
    ipMlModel   := ipMlModel,
    eKnnType    := TCVN_KNN_CLASSIFIER,
    nK          := 5,
    hrPrev      := hr);
```

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.7 F_VN_CreateLbgModel

Create a LBG model of the specified type. The initial reference count is set to one if a new model is created and kept, otherwise. Models of this type neither support on-line training (sample by sample) nor retraining.

Syntax

Definition:

```
FUNCTION F_VN_CreateLbgModel : HRESULT
VAR_INPUT
    ipMlModel         : Reference To ITcVnMlModel;
    eLbgType          : ETcVnPrototypeClusterer;
    nMaxClusters      : UDINT;
    fMaxClusterRadius : LREAL;
    bSingleSplitSteps : BOOL;
    hrPrev            : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created model (Non-zero interface pointers are reused.)
eLbgType	ETcVnPrototypeClusterer [▶ 226]	LBG model type
nMaxClusters	UDINT	Maximum number of clusters
fMaxClusterRadius	LREAL	Maximum allowed radius (L2 norm) of a single cluster, i.e. clusters with a higher radius will be split into smaller ones, until a global number of nMaxClusters is reached.
bSingleSplitSteps	BOOL	If true, the global optimization is always run after a single cluster has been split. If false, several clusters are split within the same step before applying the global optimization. Applying the global optimization less often is faster, but can lead to less optimal results, especially having 2 nearby clusters that could be represented by 1.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreateLbgModel` erstellt ein Linde-Buzo-Gray (LBG) Modell.

Linde-Buzo-Gray-Modelle

LBG-Modelle funktionieren ähnlich wie K-Means++ Modelle, indem sie iterativ die Cluster an die Samples angleichen. Die wesentliche Besonderheit von LBG ist, dass im Iterationsprozess neue Cluster angelegt werden, sobald diese größer als `fMaxClusterRadius` sind (bis die Maximalanzahl `nMaxClusters` errreicht wird). Bei K-Means++ hingegen wird die Anzahl der Cluster initial festgelegt.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Modell-Typ

`eLbgType` legt fest, ob LBG zum Clustering (`TCVN_PC_CLUSTERER`) oder zur Anomalie-Detektion (`TCVN_PC_NOVELTY_DETECTOR`) genutzt wird.

Maximale Cluster-Anzahl

Wenn durch das Aufteilen von zu großen Clustern insgesamt `nMaxClusters` existieren, werden keine neuen Cluster mehr angelegt.

Maximale Cluster-Radius

Wenn im Training der Radius eines Clusters größer als `fMaxClusterRadius` wird, wird dieser Cluster in zwei kleinere Cluster aufgeteilt.

Optimierungsqualität

Teil des iterativen Trainingsprozesses ist ein globaler Optimierungsschritt. Wenn `bSingleSplitSteps = TRUE` ist, wird diese Optimierung nach jeder einzelnen Cluster-Aufteilung ausgeführt. Dies führt zu einer höheren Qualität der Cluster, aber auch zu einer längeren Trainingszeit. Wenn `bSingleSplitSteps = FALSE` ist, wird die Optimierung erst ausgeführt, nachdem mehrere Cluster aufgeteilt wurden.

Expert-Parameter

Die Expert-Variante F_VN_CreateLbgModelExp [▶ 1426] enthält zusätzliche Parameter.

Anwendung

Ein LBG-Modell für Clustering mit `5` Clustern, einer maximalen Cluster-Größe von `3.6` und hoher Optimierungsqualität kann z.B. so erstellt werden:

```
hr := F_VN_CreateLbgModel(
    ipMlModel           := ipMlModel,
    eLbgType            := TCVN_PC_CLUSTERER,
    nMaxClusters        := 5,
    fMaxClusterRadius   := 3.6,
    bSingleSplitSteps   := TRUE,
    hrPrev              := hr);
```

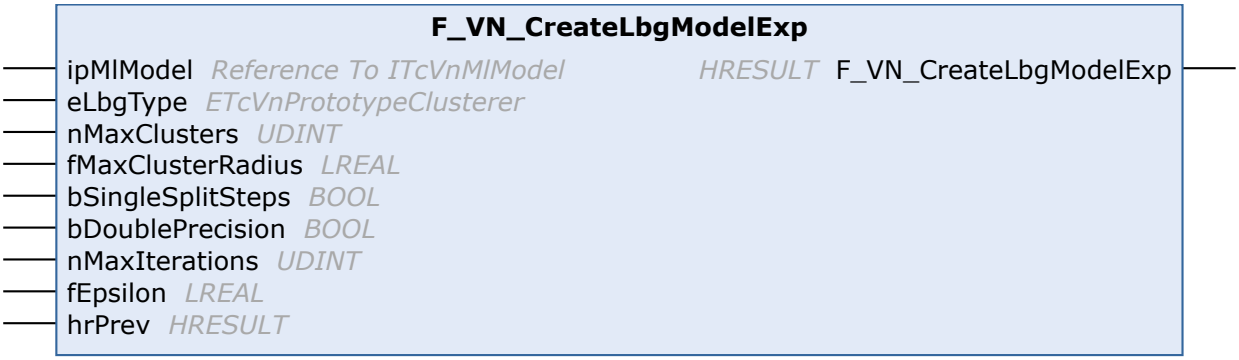
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.8 F_VN_CreateLbgModelExp



Create a LBG model of the specified type. The initial reference count is set to one if a new model is created and kept, otherwise. Models of this type neither support on-line training (sample by sample) nor retraining. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_CreateLbgModelExp : HRESULT
VAR_INPUT
    ipMlModel         : Reference To ITcVnMlModel;
    eLbgType          : ETcVnPrototypeClusterer;
    nMaxClusters      : UDINT;
    fMaxClusterRadius : LREAL;
    bSingleSplitSteps : BOOL;
    bDoublePrecision  : BOOL;
    nMaxIterations    : UDINT;
    fEpsilon          : LREAL;
    hrPrev            : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created model (Non-zero interface pointers are reused.)
eLbgType	ETcVnPrototypeClusterer [▶ 226]	LBG model type
nMaxClusters	UDINT	Maximum number of clusters
fMaxClusterRadius	LREAL	Maximum allowed radius (L2 norm) of a single cluster, i.e. clusters with a higher radius will be split into smaller ones, until a global number of nMaxClusters is reached.
bSingleSplitSteps	BOOL	If true, the global optimization is always run after a single cluster has been split. If false, several clusters are split within the same step before applying the global optimization. Applying the global optimization less often is faster, but can lead to less optimal results, especially having 2 nearby clusters that could be represented by 1.
bDoublePrecision	BOOL	If true, the model is generated with double precision (LREAL) instead of single precision (REAL). (default: FALSE)
nMaxIterations	UDINT	Maximum number of iterations (triggers the usage of the default value of 10 if it equals 0)
fEpsilon	LREAL	Maximum allowed difference of the error between two successive iterations (triggers the usage of the default value of 0.001 if it equals 0)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreateLbgModelExp` ist die Expert-Variante von F_VN_CreateLbgModel [▶ 1424]. Sie enthält zusätzliche Parameter.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Modell-Typ

`eLbgType` legt fest, ob LBG zum Clustering (`TCVN_PC_CLUSTERER`) oder zur Anomalie-Detektion (`TCVN_PC_NOVELTY_DETECTOR`) genutzt wird.

Maximale Cluster-Anzahl

Wenn durch das Aufteilen von zu großen Clustern insgesamt `nMaxClusters` existieren, werden keine neuen Cluster mehr angelegt.

Maximale Cluster-Radius

Wenn im Training der Radius eines Clusters größer als `fMaxClusterRadius` wird, wird dieser Cluster in zwei kleinere Cluster aufgeteilt.

Optimierungsqualität

Teil des iterativen Trainingsprozesses ist ein globaler Optimierungsschritt. Wenn `bSingleSplitSteps = TRUE` ist, wird diese Optimierung nach jeder einzelnen Cluster-Aufteilung ausgeführt. Dies führt zu einer höheren Qualität der Cluster, aber auch zu einer längeren Trainingszeit. Wenn `bSingleSplitSteps = FALSE` ist, wird die Optimierung erst ausgeführt, nachdem mehrere Cluster aufgeteilt wurden.

Höhere Genauigkeit

Wenn `bDoublePrecision` `TRUE` ist, wird `LREAL` als Datentyp für die internen Berechnungen des Modells genutzt; wenn es `FALSE` ist, wird `REAL` genutzt.

Maximale Iterationen

Für die Optimierung werden maximal so viele Iterationen genutzt, wie in `nMaxIterations` angegeben sind. Wenn der Wert `0` ist, wird der jeweilige Default-Wert verwendet.

Abbruchgrenze

Die Optimierung wird abgebrochen, sobald sich der Fehler zwischen zwei Iterationen nicht stärker verändert, als in `fEpsilon` angegeben ist. Wenn der Wert `0` ist, wird der jeweilige Default-Wert verwendet.

Anwendung

Ein LBG-Modell für Clustering mit `5` Clustern, einer maximalen Cluster-Größe von `3.6` und hoher Optimierungsqualität kann z.B. so erstellt werden:

```
hr := F_VN_CreateLbgModelExp(
    ipMlModel           := ipMlModel,
    eLbgType            := TCVN_PC_CLUSTERER,
    nMaxClusters        := 5,
    fMaxClusterRadius   := 3.6,
    bSingleSplitSteps   := TRUE,
    bDoublePrecision    := TRUE,
    nMaxIterations      := 0,
    fEpsilon            := 0,
    hrPrev              := hr);
```

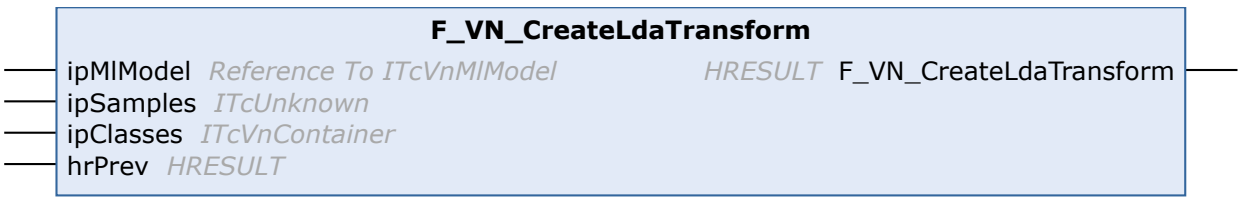
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.9 F_VN_CreateLdaTransform



Create an LDA-based feature transform from the provided data. The number of samples must be >= the number of features and the number of classes must be >= 2. The initial reference count is set to one if a new model is created and kept, otherwise.

Syntax

Definition:

```
FUNCTION F_VN_CreateLdaTransform : HRESULT
VAR_INPUT
    ipMlModel : Reference To ITcVnMlModel;
    ipSamples : ITcUnknown;
    ipClasses : ITcVnContainer;
    hrPrev    : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created feature transform (Non-zero interface pointers are reused.)
ipSamples	ITcUnknown [▶ 443]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipClasses	ITcVnContainer [▶ 385]	Class labels corresponding to the input samples (ContainerType_Vector_DINT)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreateLdaTransform` erstellt ein Linear Discriminant Analysis (LDA) Modell zur Merkmalstransformation.

Linear Discriminant Analysis-Modelle

Linear Discriminant Analysis erstellt eine Merkmalstransformation mit dem Ziel, mittels der transformierten Merkmale die Klassen möglichst einfach separierbar zu machen. Mit dem Modell lassen sich die Samples effektiv komprimieren.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Samples

Über `ipSamples` werden alle Samples in einem Container an das Modell gegeben.

Klassen

Über `ipClasses` werden die Klassenzuordnungen aller Samples in einem Container an das Modell gegeben.

Anwendung

Ein LDA-Modell zur Merkmalsreduktion kann z.B. so erstellt werden:

```
hr := F_VN_CreateLdaTransform(
    ipMlModel   := ipMlModel,
    ipSamples   := ipSamples,
    ipClasses   := ipClasses,
    hrPrev      := hr);
```

Verwandte Funktionen

- F_VN_CreateLdaTransformViaComponentNum [▶ 1430]
- F_VN_CreatePcaTransform [▶ 1433]
- F_VN_CreatePcaTransformViaComponentNum [▶ 1435]
- F_VN_CreatePcaTransformViaVariance [▶ 1436]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.10 F_VN_CreateLdaTransformViaComponentNum

Create an LDA-based feature transform from the provided data where the number of components to be used is predefined. The number of samples must be >= the number of features and the number of classes must be >= 2. The initial reference count is set to one if a new model is created and kept, otherwise. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_CreateLdaTransformViaComponentNum : HRESULT
VAR_INPUT
    ipMlModel     : Reference To ITcVnMlModel;
    ipSamples     : ITcUnknown;
    ipClasses     : ITcVnContainer;
    nComponentNum : UDINT;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created feature transform (Non-zero interface pointers are reused.)
ipSamples	ITcUnknown [▶ 443]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipClasses	ITcVnContainer [▶ 385]	Class labels corresponding to the input samples (ContainerType_Vector_DINT)
nComponentNum	UDINT	Number of components (nComponentNum must be smaller than the number of classes. A value of 0 results in the automatic computation of the maximum number of components.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreateLdaTransformViaComponentNum` ist eine Variante von F_VN_CreateLdaTransform [▶ 1428]. Sie enthält den zusätzlichen Parameter `nComponentNum`, um die Anzahl an Komponenten für die Merkmalsreduktion im Vorhinein festzulegen. Somit kann die Stärke der Datenkompression eingestellt werden.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Samples

Über `ipSamples` werden alle Samples in einem Container an das Modell gegeben.

Klassen

Über `ipClasses` werden die Klassenzuordnungen aller Samples in einem Container an das Modell gegeben.

Komponentenanzahl

Bei der Merkmalsreduktion werden `nComponentNum` Komponenten genutzt. Bei 0 wird automatisch die maximale Anzahl an Komponenten berechnet.

Anwendung

Ein LDA-Modell mit 25 Komponenten kann z.B. so erstellt werden:

```
hr := F_VN_CreateLdaTransformViaComponentNum(
    ipMlModel       := ipMlModel,
    ipSamples       := ipSamples,
    ipClasses       := ipClasses,
    nComponentNum   := 25,
    hrPrev          := hr);
```

Verwandte Funktionen

- F_VN_CreateLdaTransform [▶ 1428]
- F_VN_CreatePcaTransform [▶ 1433]

- F_VN_CreatePcaTransformViaComponentNum [▶ 1435]
- F_VN_CreatePcaTransformViaVariance [▶ 1436]

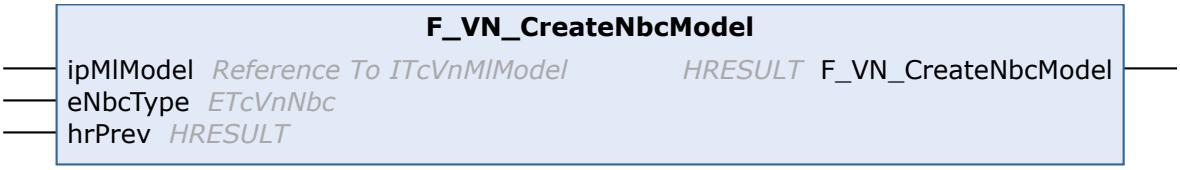
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.11 F_VN_CreateNbcModel



Create a normal Bayes classifier of the specified type. The initial reference count is set to one if a new model is created and kept, otherwise. In order to train normal Bayes classifiers, a sufficiently high number of samples is required for each class. It depends on the number of features and the distribution of the data. Hence, it needs to be tested for each application. Models of this type do not support on-line training (sample by sample). For the retraining of such classifier models, the set of presented classes must be identical to the previous learning steps. Otherwise, an exception is raised.

Syntax

Definition:

```
FUNCTION F_VN_CreateNbcModel : HRESULT
VAR_INPUT
    ipMlModel : Reference To ITcVnMlModel;
    eNbcType  : ETcVnNbc;
    hrPrev    : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created model (Non-zero interface pointers are reused.)
eNbcType	ETcVnNbc [▶ 221]	Normal Bayes classifier type
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreateNbcModel` erstellt ein Normal Bayes Classifier (NBC) Modell.

Normal Bayes Classifier-Modelle

Beim Normal Bayes Classifier wird ein Mischverteilungsmodell erstellt, bei dem jede Klasse durch eine Gaußverteilung repräsentiert wird. Während des Trainings werden die Parameter dieser Gaußfunktionen ermittelt.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Modell-Typ

`eNbcType` legt fest, ob NBC zur Klassifizierung (`TCVN_NBC_CLASSIFIER`) oder zur Anomalie-Detektion (`TCVN_NBC_NOVELTY_DETECTOR`) genutzt wird.

Anwendung

Ein NBC-Modell zur Klassifizierung kann z.B. so erstellt werden:

```
hr := F_VN_CreateNbcModel(
    ipMlModel   := ipMlModel,
    eNbcType    := TCVN_NBC_CLASSIFIER,
    hrPrev      := hr);
```

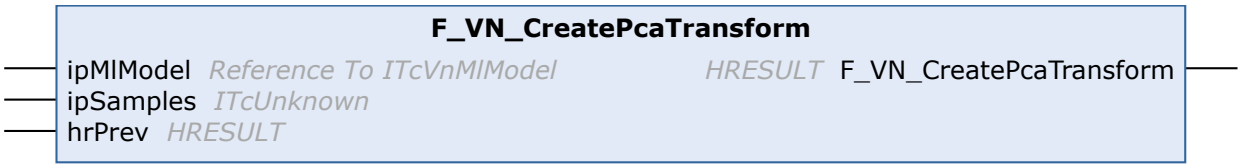
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.12 F_VN_CreatePcaTransform



Create a PCA-based feature transform from the provided data. The maximum number of principal components that can be computed equals the minimum of the number of samples and the number of features. The initial reference count is set to one if a new model is created and kept, otherwise.

Syntax

Definition:

```
FUNCTION F_VN_CreatePcaTransform : HRESULT
VAR_INPUT
    ipMlModel : Reference To ITcVnMlModel;
    ipSamples : ITcUnknown;
    hrPrev    : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created feature transform (Non-zero interface pointers are reused.)
ipSamples	ITcUnknown [▶ 443]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreatePcaTransform` erstellt ein Principal Component Analysis (PCA) Modell.

Principal Component Analysis-Modelle

Bei der Principal Component Analysis werden die Hauptkomponenten der Sample-Verteilung berechnet. Die Hauptkomponenten sind die Richtungen innerhalb des Merkmalsraums, die die größte Varianz in den Samples aufweisen. Durch eine Projektion der Daten auf ausgewählte Hauptkomponenten, bei der die Varianz der Daten bestmöglich beibehalten wird, verringert sich die Dimension des Merkmalsraums.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Samples

Über `ipSamples` werden alle Samples in einem Container an das Modell gegeben.

Anwendung

Ein PCA-Modell zur Merkmalsreduktion kann z.B. so erstellt werden:

```
hr := F_VN_CreatePcaTransform(
    ipMlModel   := ipMlModel,
    ipSamples   := ipSamples,
    hrPrev      := hr);
```

Verwandte Funktionen

- F_VN_CreatePcaTransformViaComponentNum [▶ 1435]
- F_VN_CreatePcaTransformViaVariance [▶ 1436]
- F_VN_CreateLdaTransform [▶ 1428]
- F_VN_CreateLdaTransformViaComponentNum [▶ 1430]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.13 F_VN_CreatePcaTransformViaComponentNum

Create a PCA-based feature transform from the provided data where the number of prinicpal components to be used is predefined. The maximum number of principal components that can be computed equals the minimum of the number of samples and the number of features. The initial reference count is set to one if a new model is created and kept, otherwise. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_CreatePcaTransformViaComponentNum : HRESULT
VAR_INPUT
    ipMlModel     : Reference To ITcVnMlModel;
    ipSamples     : ITcUnknown;
    nComponentNum : UDINT;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created feature transform (Non-zero interface pointers are reused.)
ipSamples	ITcUnknown [▶ 443]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
nComponentNum	UDINT	Number of principal components (A value of 0 results in the automatic computation of the maximum number of components.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreatePcaTransformViaComponentNum` ist eine Variante von F_VN_CreatePcaTransform [▶ 1433]. Sie enthält den zusätzlichen Parameter `nComponentNum`, um die Anzahl an Hauptkomponenten für die Merkmalsreduktion im Vorhinein festzulegen.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Samples

Über `ipSamples` werden alle Samples in einem Container an das Modell gegeben.

Komponentenanzahl

Bei der PCA werden `nComponentNum` Hauptkomponenten berechnet. Bei 0 wird automatisch die maximale Anzahl an Komponenten berechnet.

Anwendung

Ein PCA-Modell mit 5 Hauptkomponenten kann z.B. so erstellt werden:

```
hr := F_VN_CreatePcaTransformViaComponentNum(
    ipMlModel       := ipMlModel,
    ipSamples       := ipSamples,
    nComponentNum   := 5,
    hrPrev          := hr);
```

Verwandte Funktionen

- F_VN_CreatePcaTransform [▶ 1433]
- F_VN_CreatePcaTransformViaVariance [▶ 1436]
- F_VN_CreateLdaTransform [▶ 1428]
- F_VN_CreateLdaTransformViaComponentNum [▶ 1430]

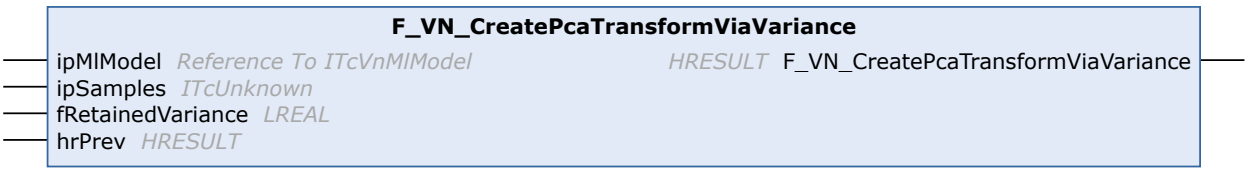
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.14 F_VN_CreatePcaTransformViaVariance



Create a PCA-based feature transform from the provided data based on a given fraction of variance to be retained. The maximum number of principal components that can be computed equals the minimum of the number of samples and the number of features. The initial reference count is set to one if a new model is created and kept, otherwise. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_CreatePcaTransformViaVariance : HRESULT
VAR_INPUT
    ipMlModel         : Reference To ITcVnMlModel;
    ipSamples         : ITcUnknown;
    fRetainedVariance : LREAL;
    hrPrev            : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created feature transform (Non-zero interface pointers are reused.)
ipSamples	ITcUnknown [▶ 443]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
fRetainedVariance	LREAL	Fraction of variance that is to be retained by the PCA (A value of 1.0 signifies 100%.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreatePcaTransformViaVariance` ist eine Variante von F_VN_CreatePcaTransform [▶ 1433]. Sie enthält den zusätzlichen Parameter `fRetainedVariance`, um die verbleibende Varianz festzulegen.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Samples

Über `ipSamples` werden alle Samples in einem Container an das Modell gegeben.

Restliche Varianz

Bei der PCA werden die Hauptkomponenten so ausgewählt, dass der Anteil `fRetainedVariance` der Varianz in den Samples bestehen bleibt. Bei `1` bleibt die gesamte Varianz bestehen, sodass die Anzahl der Merkmale nicht reduziert wird.

Anwendung

Ein PCA-Modell mit 65% verbleibender Varianz kann z.B. so erstellt werden:

```
hr := F_VN_CreatePcaTransformViaComponentNum(
    ipMlModel           := ipMlModel,
    ipSamples           := ipSamples,
    fRetainedVariance   := 0.65,
    hrPrev              := hr);
```

Verwandte Funktionen

- F_VN_CreatePcaTransform [▶ 1433]
- F_VN_CreatePcaTransformViaComponentNum [▶ 1435]
- F_VN_CreateLdaTransform [▶ 1428]
- F_VN_CreateLdaTransformViaComponentNum [▶ 1430]

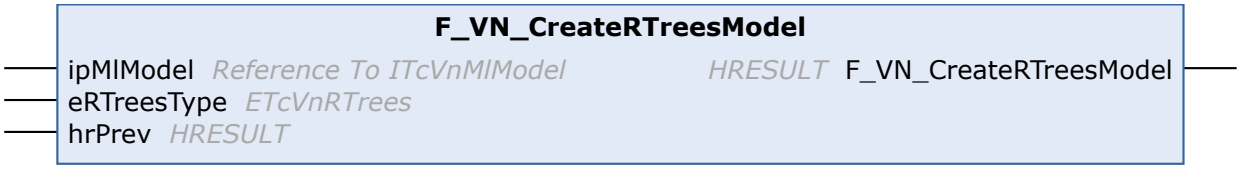
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.15 F_VN_CreateRTreesModel



Create a random trees model of the specified type. The initial reference count is set to one if a new model is created and kept, otherwise. Models of this type neither support on-line training (sample by sample) nor retraining. Predictions can only be scalar.

Syntax

Definition:

```
FUNCTION F_VN_CreateRTreesModel : HRESULT
VAR_INPUT
    ipMlModel   : Reference To ITcVnMlModel;
    eRTreesType : ETcVnRTrees;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created model (Non-zero interface pointers are reused.)
eRTreesType	ETcVnRTrees [▶ 227]	Random trees model type
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreateRTreesModel` erstellt ein Random Forest (RTrees) Modell.

Random Forest-Modelle

Das Random Forest Modell nutzt Entscheidungsbäume, deren Verzweigungen durch einen Zufallsprozess festgelegt werden. Die Prädiktion der RTrees-Modelle wird aus den Ergebnissen der einzelnen Entscheidungsbäume ermittelt.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Modell-Typ

`eRTreesType` legt fest, ob das Random Forest-Modell zur Klassifizierung (`TCVN_RT_CLASSIFIER`) oder zur Regression (`TCVN_RT_REGRESSOR`) genutzt wird.

Expert-Parameter

Die Expert-Varianten F_VN_CreateRTreesModelExp [▶ 1439] und F_VN_CreateRTreesModelExp2 [▶ 1441] enthalten zusätzliche Parameter.

Anwendung

Ein RTrees-Modell zur Regression kann z.B. so erstellt werden:

```
hr := F_VN_CreateRTreesModel(
    ipMlModel   := ipMlModel,
    eRTreesType := TCVN_RT_REGRESSOR,
    hrPrev      := hr);
```

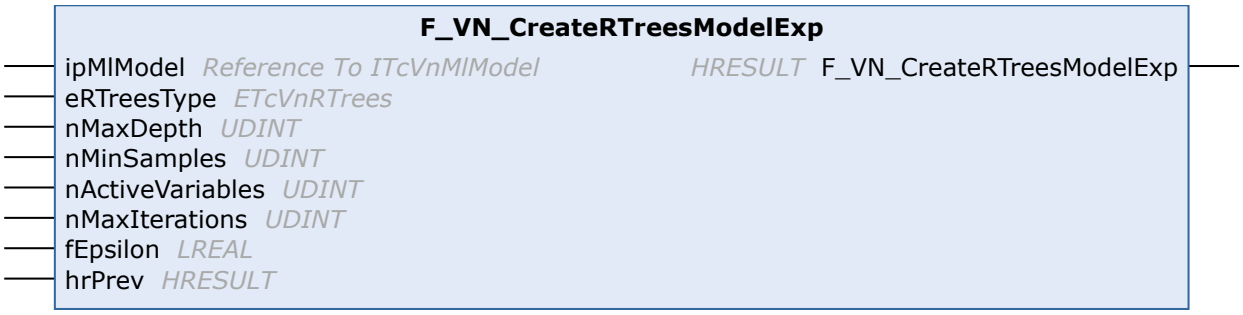
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.16 F_VN_CreateRTreesModelExp



Create a random trees model of the specified type. The initial reference count is set to one if a new model is created and kept, otherwise. Models of this type neither support on-line training (sample by sample) nor retraining. Predictions can only be scalar. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_CreateRTreesModelExp : HRESULT
VAR_INPUT
    ipMlModel        : Reference To ITcVnMlModel;
    eRTreesType      : ETcVnRTrees;
    nMaxDepth        : UDINT;
    nMinSamples      : UDINT;
    nActiveVariables : UDINT;
    nMaxIterations   : UDINT;
    fEpsilon         : LREAL;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created model (Non-zero interface pointers are reused.)
eRTreesType	ETcVnRTrees [▶ 227]	Random trees model type
nMaxDepth	UDINT	Maximum tree depth (default: 5)
nMinSamples	UDINT	Minimum number of samples within a node required for splitting (default: 10)
nActiveVariables	UDINT	Number of variables considered for splitting (0 means sqrt(total number of variables); default: 0)
nMaxIterations	UDINT	Maximum number of iterations (disabled if it equals 0 and fEpsilon is different from 0.0; triggers the usage of the default value of 50 if nMaxIterations and fEpsilon equal 0)
fEpsilon	LREAL	Maximum allowed difference of the error between two successive iterations (disabled if it equals 0.0 and nMaxIterations is different from 0; triggers the usage of the default value of 0.1 if nMaxIterations and fEpsilon equal 0)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreateRTReesModelExp` ist eine Expert-Variante von F_VN_CreateRTReesModel [▶ 1438]. Sie enthält zusätzliche Parameter.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Modell-Typ

`eRTreesType` legt fest, ob das Random Forest-Modell zur Klassifizierung (`TCVN_RT_CLASSIFIER`) oder zur Regression (`TCVN_RT_REGRESSOR`) genutzt wird.

Maximale Baumtiefe

`nMaxDepth` gibt an, wie viele Entscheidungsebenen es maximal in einem Baum gibt.

Minimale Sample-Anzahl innerhalb eines Knotens

`nMinSamples` ist die minimale Anzahl an Samples, die während der Baumbildung in einem Knoten verbleiben muss. Aufspaltungen, die diese Anzahl unterschreiten würden, werden nicht durchgeführt.

Splitting-Merkmale

`nActiveVariables` ist die Anzahl an Merkmalen, die zum Aufteilen des Baumes genutzt werden. Bei `0` werden `sqrt(N)` Merkmale mit der Anzahl `N` aller verfügbaren Merkmale genutzt.

Maximale Iterationen

Für die Optimierung werden maximal so viele Iterationen genutzt, wie in `nMaxIterations` angegeben sind. Wenn der Wert `0` ist, wird der jeweilige Default-Wert verwendet.

Abbruchgrenze

Die Optimierung wird abgebrochen, sobald sich der Fehler zwischen zwei Iterationen nicht stärker verändert, als in `fEpsilon` angegeben ist. Wenn der Wert `0` ist, wird der jeweilige Default-Wert verwendet.

Anwendung

Ein RTrees-Modell zur Regression kann z.B. so erstellt werden:

```
hr := F_VN_CreateRTreesModelExp(
    ipMlModel           := ipMlModel,
    eRTreesType         := TCVN_RT_REGRESSOR,
    nMaxDepth           := 8,
    nMinSamples         := 4,
    nActiveVariables    := 0,
    nMaxIterations      := 0,
    fEpsilon            := 0,
    hrPrev              := hr);
```

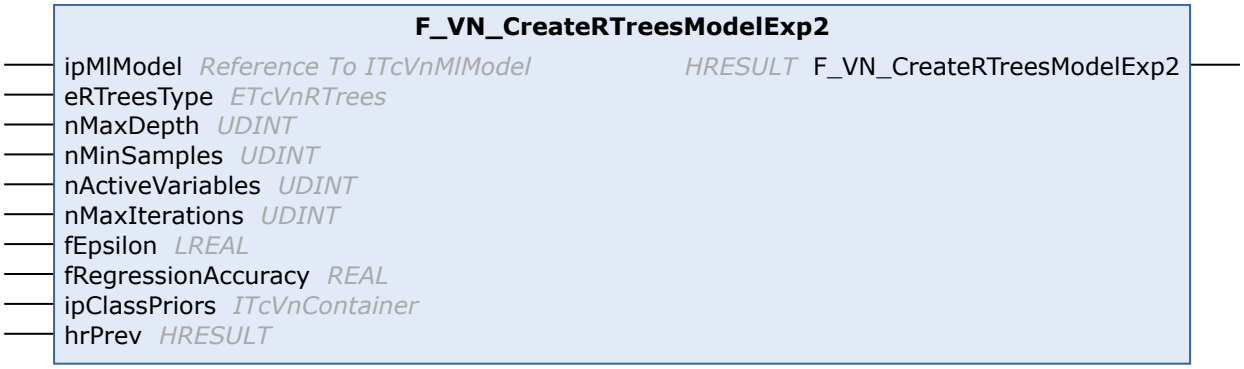
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.17 F_VN_CreateRTreesModelExp2



Create a random trees model of the specified type. The initial reference count is set to one if a new model is created and kept, otherwise. Models of this type neither support on-line training (sample by sample) nor retraining. Predictions can only be scalar. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_CreateRTreesModelExp2 : HRESULT
VAR_INPUT
    ipMlModel           : Reference To ITcVnMlModel;
    eRTreesType         : ETcVnRTrees;
    nMaxDepth           : UDINT;
    nMinSamples         : UDINT;
    nActiveVariables    : UDINT;
    nMaxIterations      : UDINT;
    fEpsilon            : LREAL;
    fRegressionAccuracy : REAL;
    ipClassPriors       : ITcVnContainer;
    hrPrev              : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created model (Non-zero interface pointers are reused.)
eRTreesType	ETcVnRTrees [▶ 227]	Random trees model type
nMaxDepth	UDINT	Maximum tree depth (default: 5)
nMinSamples	UDINT	Minimum number of samples within a node required for splitting (default: 10)
nActiveVariables	UDINT	Number of variables considered for splitting (0 means sqrt(total number of variables); default: 0)
nMaxIterations	UDINT	Maximum number of iterations (disabled if it equals 0 and fEpsilon is different from 0.0; triggers the usage of the default value of 50 if nMaxIterations and fEpsilon equal 0)
fEpsilon	LREAL	Maximum allowed difference of the error between two successive iterations (disabled if it equals 0.0 and nMaxIterations is different from 0; triggers the usage of the default value of 0.1 if nMaxIterations and fEpsilon equal 0)
fRegressionAccuracy	REAL	Termination criterion for regressors (only for regressors; set to default if not allowed; default: 0.0)
ipClassPriors	ITcVnContainer [▶ 385]	Class priors (ContainerType_Vector_REAL or ContainerType_Vector_LREAL; only for classifiers; optional, set to 0 if not required or not allowed; default: 0)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreateRTReesModelExp2` ist eine Expert-Variante von F_VN_CreateRTreesModel [▶ 1438]. Sie enthält zusätzliche Parameter.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Modell-Typ

`eRTreesType` legt fest, ob das Random Forest-Modell zur Klassifizierung (`TCVN_RT_CLASSIFIER`) oder zur Regression (`TCVN_RT_REGRESSOR`) genutzt wird.

Maximale Baumtiefe

`nMaxDepth` gibt an, wie viele Entscheidungsebenen es maximal in einem Baum gibt.

Minimale Sample-Anzahl innerhalb eines Knotens

`nMinSamples` ist die minimale Anzahl an Samples, die während der Baumbildung in einem Knoten verbleiben muss. Aufspaltungen, die diese Anzahl unterschreiten würden, werden nicht durchgeführt.

Splitting-Merkmale

`nActiveVariables` ist die Anzahl an Merkmalen, die zum Aufteilen des Baumes genutzt werden. Bei `0` werden `sqrt(N)` Merkmale mit der Anzahl `N` aller verfügbaren Merkmale genutzt.

Maximale Iterationen

Für die Optimierung werden maximal so viele Iterationen genutzt, wie in `nMaxIterations` angegeben sind. Wenn der Wert `0` ist, wird der jeweilige Default-Wert verwendet.

Abbruchgrenze

Die Optimierung wird abgebrochen, sobald sich der Fehler zwischen zwei Iterationen nicht stärker verändert, als in `fEpsilon` angegeben ist. Wenn der Wert `0` ist, wird der jeweilige Default-Wert verwendet.

Regressionsgenauigkeit

`fRegressionAccuracy` ist das Abbruchkriterium für Regressionsoptimierung. Bei einer Klassifizierung muss der Wert auf `0` gesetzt werden.

Class-Priors

`ipClassPriors` ist ein Container, mit dem die Apriori-Wahrscheinlichkeiten der einzelnen Klassen angegeben werden können. Bei einer Regression sollte der Wert auf `0` gesetzt werden.

Anwendung

Ein RTrees-Modell zur Regression kann z.B. so erstellt werden:

```
hr := F_VN_CreateRTreesExp2(
    ipMlModel           := ipMlModel,
    eRTreesType         := TCVN_RT_REGRESSOR,
    nMaxDepth           := 8,
    nMinSamples         := 4,
    nActiveVariables    := 0,
    nMaxIterations      := 0,
    fEpsilon            := 0,
    fRegressionAccuracy := 0.3,
    ipClassPriors       := 0,
    hrPrev              := hr);
```

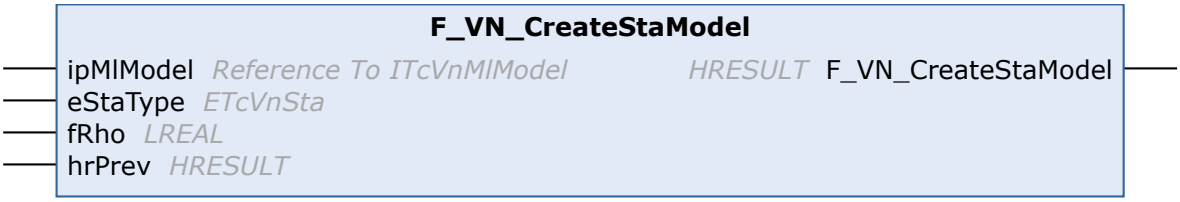
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.18 F_VN_CreateStaModel



Create a Simplified TopoART neural network of the specified type. The initial reference count is set to one if a new model is created and kept, otherwise. Models of this type support on-line training (sample by sample), retraining, as well as scalar and vectorial predictions. It requires all input except class labels (i.e., samples and training outputs/predictions) to lie in the interval [0.0, 1.0]. The predictions of regressors need to be rescaled from the interval [0.0, 1.0] to their respective value range before usage. Depending on the parameter settings and the number of available training samples, repeated training with the same data may improve the results. Like other neural networks based on the Adaptive Resonance Theory (ART), Simplified TopoART neural networks are not prone to catastrophic inference and patricularly well-suited to incremental learning tasks.

Syntax

Definition:

```
FUNCTION F_VN_CreateStaModel : HRESULT
VAR_INPUT
    ipMlModel : Reference To ITcVnMlModel;
    eStaType  : ETcVnSta;
    fRho      : LREAL;
    hrPrev    : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created model (Non-zero interface pointers are reused.)
eStaType	ETcVnSta [▶ 229]	Simpified TopoART model type
fRho	LREAL	Vigilance parameter (controls the number of neurons that are inserted and the maximum size of the formed categories; valid range: [0.0, 1.0]; suggested range: [0.8, 1.0))
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreateStaModel` erstellt ein Simplified TopoART (STA) Modell.

Simplified TopoART-Modelle

Das Simplified TopoART Netzwerk lernt Teilbereiche des Merkmalsraums. Je nach Anwendung werden diese zu Clustern zusammengefasst, Klassen zugeordnet oder für die Prädiktion verwendet. Aufgrund der inkrementellen Architektur ist ein Nachtraining möglich, wobei die bereits gelernten Informationen bestmöglich erhalten werden. Als Eingaben (mit Ausnahme der Klassenbezeichnungen) werden nur Werte im Intervall von 0.0 und 1.0 akzeptiert. Daher empfiehlt es sich eine Merkmalsnormierung zu verwenden.

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Modell-Typ

`eStaType` legt fest, ob das STA-Modell zur Klassifizierung (`TCVN_STA_CLASSIFIER`), zur Regression (`TCVN_STA_REGRESSOR`), zum Clustering (`TCVN_STA_CLUSTERER`) oder zur Anomalie-Detektion (`TCVN_STA_NOVELTY_DETECTOR`) genutzt wird.

Vigilanz-Parameter

Zur Anpassung des Modells an die jeweilige Aufgabenstellung ist dieser Parameter entscheidend, da er die maximale Größe der zu lernenden Teilbereiche steuert.

Expert-Parameter

Die Expert-Varianten F_VN_CreateStaModelExp [▶ 1445] und F_VN_CreateStaModelExp2 [▶ 1447] enthalten zusätzliche Parameter.

Anwendung

Ein STA-Modell zur Klassifikation kann z.B. so erstellt werden:

```
hr := F_VN_CreateStaModel(
    ipMlModel    := ipMlModel,
    eStaType     := TCVN_STA_CLASSIFIER,
    fRho         := 0.9,
    hrPrev       := hr);
```

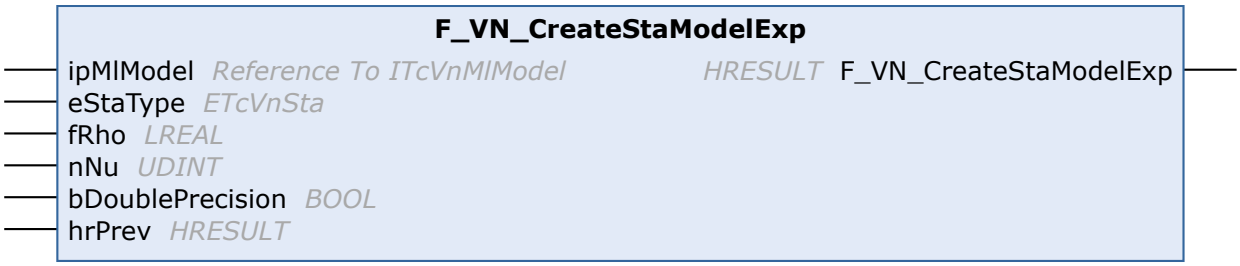
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.19 F_VN_CreateStaModelExp



Create a Simplified TopoART neural network of the specified type. The initial reference count is set to one if a new model is created and kept, otherwise. Models of this type support on-line training (sample by sample), retraining, as well as scalar and vectorial predictions. It requires all input except class labels (i.e., samples and training outputs/predictions) to lie in the interval [0.0, 1.0]. The predictions of regressors need to be rescaled from the interval [0.0, 1.0] to their respective value range before usage. Depending on the parameter settings and the number of available training samples, repeated training with the same data may improve the results. Like other neural networks based on the Adaptive Resonance Theory (ART), Simplified TopoART neural networks are not prone to catastrophic inference and patricularly well-suited to incremental learning tasks. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_CreateStaModelExp : HRESULT
VAR_INPUT
    ipMlModel        : Reference To ITcVnMlModel;
    eStaType         : ETcVnSta;
    fRho             : LREAL;
    nNu              : UDINT;
    bDoublePrecision : BOOL;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created model (Non-zero interface pointers are reused.)
eStaType	ETcVnSta [▶ 229]	Simpified TopoART model type
fRho	LREAL	Vigilance parameter (controls the number of neurons that are inserted and the maximum size of the formed categories; valid range: [0.0, 1.0]; suggested range: [0.8, 1.0))
nNu	UDINT	Number of neurons used for classification and prediction (must be larger than or equal to 1; suggested range: [1, 10); default: 3)
bDoublePrecision	BOOL	If true, the model is generated with double precision (LREAL) instead of single precision (REAL). (default: FALSE)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreateStaModelExp` ist eine Expert-Variante von F_VN_CreateStaModel [▶ 1443]. Sie enthält zusätzliche Parameter.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Modell-Typ

`eStaType` legt fest, ob das STA-Modell zur Klassifizierung (`TCVN_STA_CLASSIFIER`), zur Regression (`TCVN_STA_REGRESSOR`), zum Clustering (`TCVN_STA_CLUSTERER`) oder zur Anomalie-Detektion (`TCVN_STA_NOVELTY_DETECTOR`) genutzt wird.

Vigilanz-Parameter

Zur Anpassung des Modells an die jeweilige Aufgabenstellung ist dieser Parameter entscheidend, da er die maximale Größe der zu lernenden Teilbereiche steuert.

Anzahl genutzter Nachbarn

`nNu` gibt die Anzahl der für die Klassifizierung und Vorhersage verwendeten Nachbarteilbereiche an und muss >= 1 sein.

Höhere Genauigkeit

Wenn `bDoublePrecision` `TRUE` ist, wird `LREAL` als Datentyp für die internen Berechnungen des Modells genutzt; wenn es `FALSE` ist, wird `REAL` genutzt.

Anwendung

Ein STA-Modell zur Klassifikation kann z.B. so erstellt werden:

```
hr := F_VN_CreateStaModel(
    ipMlModel        := ipMlModel,
    eStaType         := TCVN_STA_CLASSIFIER,
    fRho             := 0.9,
    nNu              := 3,
    bDoublePrecision := FALSE,
    hrPrev           := hr);
```

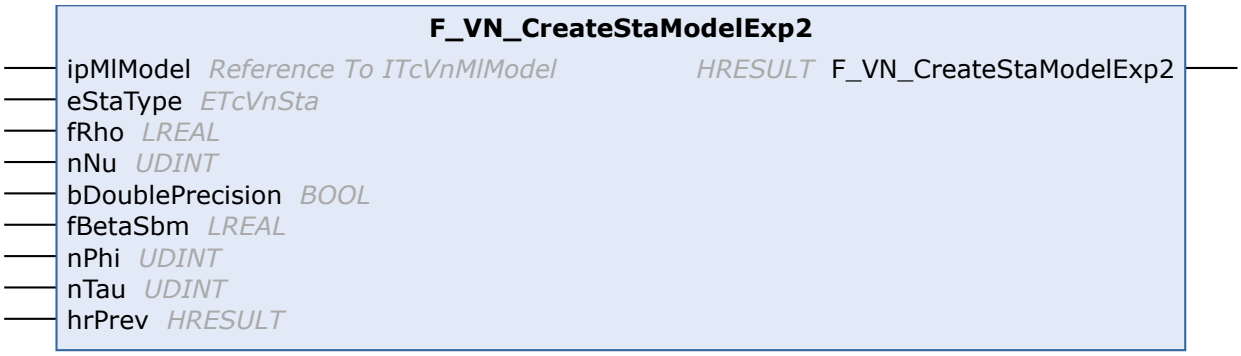
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.20 F_VN_CreateStaModelExp2



Create a Simplified TopoART neural network of the specified type. The initial reference count is set to one if a new model is created and kept, otherwise. Models of this type support on-line training (sample by sample), retraining, as well as scalar and vectorial predictions. It requires all input except class labels (i.e., samples and training outputs/predictions) to lie in the interval [0.0, 1.0]. The predictions of regressors need to be rescaled from the interval [0.0, 1.0] to their respective value range before usage. Depending on the parameter settings and the number of available training samples, repeated training with the same data may improve the results. Like other neural networks based on the Adaptive Resonance Theory (ART), Simplified TopoART neural networks are not prone to catastrophic inference and patricularly well-suited to incremental learning tasks. (additional expert function providing parameters for fine-tuning and noise reduction)

Syntax

Definition:

```
FUNCTION F_VN_CreateStaModelExp2 : HRESULT
VAR_INPUT
    ipMlModel        : Reference To ITcVnMlModel;
    eStaType         : ETcVnSta;
    fRho             : LREAL;
    nNu              : UDINT;
    bDoublePrecision : BOOL;
    fBetaSbm         : LREAL;
    nPhi             : UDINT;
    nTau             : UDINT;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created model (Non-zero interface pointers are reused.)
eStaType	ETcVnSta [▶ 229]	Simpified TopoART model type
fRho	LREAL	Vigilance parameter (controls the number of neurons that are inserted and the maximum size of the formed categories; valid range: [0.0, 1.0]; suggested range: [0.8, 1.0))
nNu	UDINT	Number of neurons used for classification and prediction (must be larger than or equal to 1; suggested range: [1, 10); default: 3)
bDoublePrecision	BOOL	If true, the model is generated with double precision (LREAL) instead of single precision (REAL). (default: FALSE)
fBetaSbm	LREAL	Learning rate of the second best-matching neuron (learning the second best-matching neuron keeps related categories closer together and might improve the results but may require a higher number of neurons and training runs; a value of 0.0 disables learning of the second best-matching neuron; valid range: [0.0, 1.0]; suggested range: [0.0, 0.5]); default: 0.0
nPhi	UDINT	Number of samples a neuron must have learnt to become permanent (required for noise reduction; works in conjunction with nTau; must be larger than or equal to 1; higher values intensify noise reduction; a value of 1 disables removal of neuron candidates; default: 1)
nTau	UDINT	Number of learning steps after which node removal is performed (required for noise reduction; works in conjunction with nPhi; must be large enough to allow neuron candidates representing non-noise samples to become the best-matching neuron at least nPhi times before node removal; default: 100)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreateStaModelExp2` ist eine Expert-Variante von F_VN_CreateStaModel [▶ 1443]. Sie enthält zusätzliche Parameter.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Modell-Typ

`eStaType` legt fest, ob das STA-Modell zur Klassifizierung (`TCVN_STA_CLASSIFIER`), zur Regression (`TCVN_STA_REGRESSOR`), zum Clustering (`TCVN_STA_CLUSTERER`) oder zur Anomalie-Detektion (`TCVN_STA_NOVELTY_DETECTOR`) genutzt wird.

Vigilanz-Parameter

Zur Anpassung des Modells an die jeweilige Aufgabenstellung ist dieser Parameter entscheidend, da er die maximale Größe der zu lernenden Teilbereiche steuert.

Anzahl genutzter Nachbarn

`nNu` gibt die Anzahl der für die Klassifizierung und Vorhersage verwendeten Nachbarteilbereiche an und muss >= 1 sein.

Höhere Genauigkeit

Wenn `bDoublePrecision` `TRUE` ist, wird `LREAL` als Datentyp für die internen Berechnungen des Modells genutzt; wenn es `FALSE` ist, wird `REAL` genutzt.

Anpassung der Verteilung

`fBetaSbm` ergänzt die Steuerung zur Anpassung der Teilbereiche an die zugrundeliegende Sample-Verteilung.

Rauschunterdrückung

Mit `nPhi` und `nTau` wird die Unterdrückung von sehr seltenen Samples (Rauschen, Ausreißer) im Lernprozess gesteuert.

`nPhi` gibt die Anzahl der Samples an, die in einem Teilbereich gelernt werden müssen, damit dieser Teilbereich dauerhaft berücksichtigt wird.

`nTau` gibt an, nach wie vielen Lernschritten die Bedingung von `nPhi` geprüft wird und Teilbereiche, die diese Bedingung nicht erfüllen, entfernt werden.

Erhöhen von `nPhi` und verringern von `nTau` verstärken die Rauschunterdrückung, wobei das Verhältnis von `nPhi` zu `nTau` als Maß zur Abschätzung der Stärke genutzt werden kann.

Anwendung

Ein STA-Modell zur Klassifikation kann z.B. so erstellt werden:

```
hr := F_VN_CreateStaModel(
    ipMlModel        := ipMlModel,
    eStaType         := TCVN_STA_CLASSIFIER,
    fRho             := 0.9,
    nNu              := 3,
    bDoublePrecision := FALSE,
    fBetaSbm         := 0.0,
    nPhi             := 1,
    nTau             := 100,
    hrPrev           := hr);
```

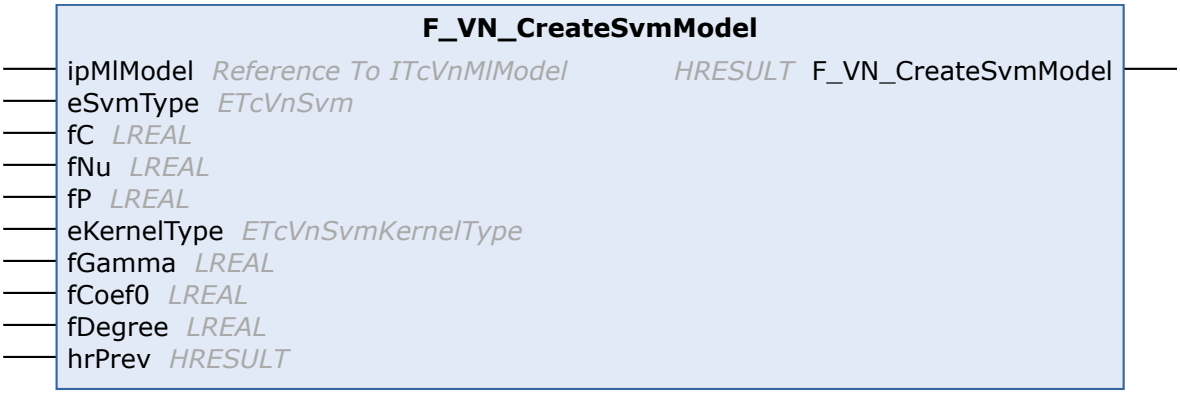
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.21 F_VN_CreateSvmModel



Create an SVM model of the specified type. The initial reference count is set to one if a new model is created and kept, otherwise. Models of this type neither support on-line training (sample by sample) nor retraining. Predictions can only be scalar.

Syntax

Definition:

```
FUNCTION F_VN_CreateSvmModel : HRESULT
VAR_INPUT
    ipMlModel   : Reference To ITcVnMlModel;
    eSvmType    : ETcVnSvm;
    fC          : LREAL;
    fNu         : LREAL;
    fP          : LREAL;
    eKernelType : ETcVnSvmKernelType;
    fGamma      : LREAL;
    fCoef0      : LREAL;
    fDegree     : LREAL;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created model (Non-zero interface pointers are reused.)
eSvmType	ETcVnSvm [▶ 230]	SVM model type
fC	LREAL	Parameter C (required for TCVN_SVM_C_CLASSIFIER, TCVN_SVM_EPS_REGRESSOR, and TCVN_SVM_NU_REGRESSOR; ignored otherwise)
fNu	LREAL	Parameter nu (required for TCVN_SVM_NU_CLASSIFIER, TCVN_SVM_NOVELTY_DETECTOR, and TCVN_SVM_NU_REGRESSOR; ignored otherwise)
fP	LREAL	Parameter p (required for TCVN_SVM_EPS_REGRESSOR; ignored otherwise)
eKernelType	ETcVnSvmKernelType [▶ 230]	Kernel type
fGamma	LREAL	Parameter gamma (used by polynomial, RBF, sigmoid, and chi-squared kernels; ignored otherwise)
fCoef0	LREAL	parameter coef0 (used by polynomial and sigmoid kernels; ignored otherwise)
fDegree	LREAL	Degree (used by polynomial kernels; ignored otherwise)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreateSvmModel` erstellt ein Support Vector Machine (SVM) Modell.

Support Vector Machine-Modelle

Support Vector Maschinen können auch für komplexe nichtlineare Problemstellungen verwendet werden. Die Prädiktion stützt sich ausschließlich auf die sogenannten Support-Vektoren, weshalb SVM-Modelle vergleichsweise speichereffizient sind. Für das Training werden alle Samples gleichzeitig benötigt (Batch-Training) und ein Nachtrainieren ist nicht möglich.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Modell-Typ

`eSvmType` legt fest, ob das SVM-Modell zur Klassifizierung, zur Regression oder zur Anomalie-Detektion genutzt wird:

- TCVN_SVM_C_CLASSIFIER
- TCVN_SVM_NU_CLASSIFIER
- TCVN_SVM_NOVELTY_DETECTOR
- TCVN_SVM_EPS_REGRESSOR
- TCVN_SVM_NU_REGRESSOR

Modell-Parameter

Die Verwendung und Bedeutung der Parameter `fC`, `fNu` und `fP` hängt vom ausgewähltem `eSvmType` ab.

Kernel-Typ

Mit `eKernelType` wird der verwendete Kernel-Typ zur Berechnung festgelegt. Der Kernel-Typ hängt von der Aufgabenstellung / Datenverteilung ab und muss daran angepasst werden.

Kernel-Parameter

Die Verwendung und Bedeutung der Parameter `fGamma`, `fCoef0` und `fDegree` hängt vom ausgewähltem `eKernelType` ab.

Expert-Parameter

Die Expert-Varianten F_VN_CreateSvmModelExp [▶ 1452] und F_VN_CreateSvmModelExp2 [▶ 1455] enthalten zusätzliche Parameter.

Anwendung

Ein SVM-Modell zur Klassifikation kann z.B. so erstellt werden:

```
hr := F_VN_CreateSvmModel(
    ipMlModel   := ipMlModel,
    eSvmType    := TCVN_SVM_NU_CLASSIFIER,
    fC          := 0,
    fNu         := 0.1,
    fP          := 0,
    eKernelType := TCVN_SKT_RBF,
    fGamma      := 1,
    fCoef0      := 0,
    fDegree     := 0,
    hrPrev      := hr);
```

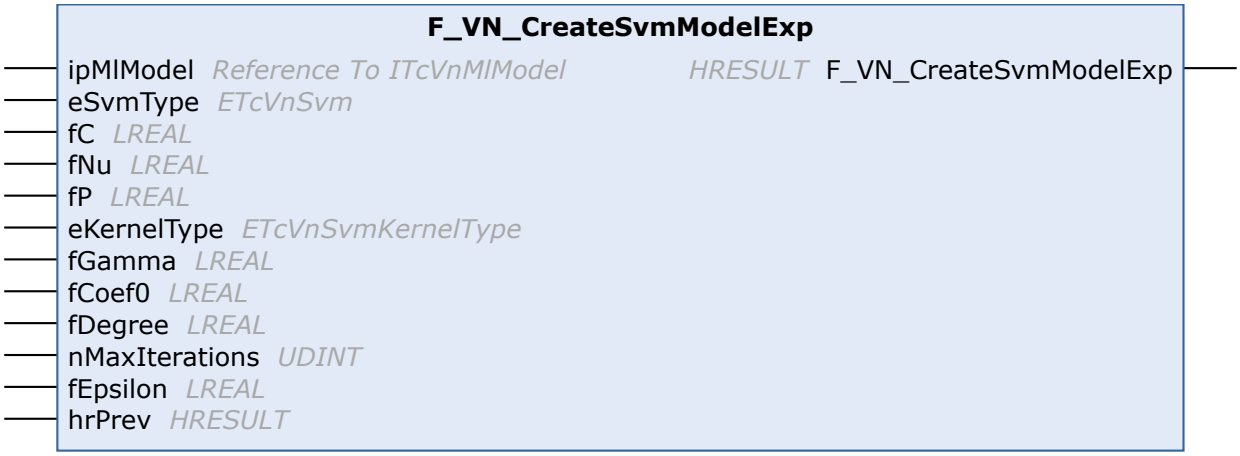
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.22 F_VN_CreateSvmModelExp



Create an SVM model of the specified type. The initial reference count is set to one if a new model is created and kept, otherwise. Models of this type neither support on-line training (sample by sample) nor retraining. Predictions can only be scalar. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_CreateSvmModelExp : HRESULT
VAR_INPUT
    ipMlModel      : Reference To ITcVnMlModel;
    eSvmType       : ETcVnSvm;
    fC             : LREAL;
    fNu            : LREAL;
    fP             : LREAL;
    eKernelType    : ETcVnSvmKernelType;
    fGamma         : LREAL;
    fCoef0         : LREAL;
    fDegree        : LREAL;
    nMaxIterations : UDINT;
    fEpsilon       : LREAL;
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created model (Non-zero interface pointers are reused.)
eSvmType	ETcVnSvm [▶ 230]	SVM model type
fC	LREAL	Parameter C (required for TCVN_SVM_C_CLASSIFIER, TCVN_SVM_EPS_REGRESSOR, and TCVN_SVM_NU_REGRESSOR; ignored otherwise)
fNu	LREAL	Parameter nu (required for TCVN_SVM_NU_CLASSIFIER, TCVN_SVM_NOVELTY_DETECTOR, and TCVN_SVM_NU_REGRESSOR; ignored otherwise)
fP	LREAL	Parameter p (required for TCVN_SVM_EPS_REGRESSOR; ignored otherwise)
eKernelType	ETcVnSvmKernelType [▶ 230]	Kernel type
fGamma	LREAL	Parameter gamma (used by polynomial, RBF, sigmoid, and chi-squared kernels; ignored otherwise)
fCoef0	LREAL	Parameter coef0 (used by polynomial and sigmoid kernels; ignored otherwise)
fDegree	LREAL	Degree (used by polynomial kernels; ignored otherwise)
nMaxIterations	UDINT	Maximum number of iterations (disabled if it equals 0 and fEpsilon is different from 0.0; triggers the usage of the default value of 100000 if nMaxIterations and fEpsilon equal 0)
fEpsilon	LREAL	Maximum allowed difference of the error between two successive iterations (disabled if it equals 0.0 and nMaxIterations is different from 0; triggers the usage of the default value of 0.00001 if nMaxIterations and fEpsilon equal 0)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreateSvmModelExp` ist eine Expert-Variante von F_VN_CreateSvmModel [▶ 1449]. Sie enthält zusätzliche Parameter.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Modell-Typ

`eSvmType` legt fest, ob das SVM-Modell zur Klassifizierung, zur Regression oder zur Anomalie-Detektion genutzt wird:

- TCVN_SVM_C_CLASSIFIER
- TCVN_SVM_NU_CLASSIFIER
- TCVN_SVM_NOVELTY_DETECTOR
- TCVN_SVM_EPS_REGRESSOR
- TCVN_SVM_NU_REGRESSOR

Modell-Parameter

Die Verwendung und Bedeutung der Parameter `fC`, `fNu` und `fP` hängt vom ausgewähltem `eSvmType` ab.

Kernel-Typ

Mit `eKernelType` wird der verwendete Kernel-Typ zur Berechnung festgelegt. Der Kernel-Typ hängt von der Aufgabenstellung / Datenverteilung ab und muss daran angepasst werden.

Kernel-Parameter

Die Verwendung und Bedeutung der Parameter `fGamma`, `fCoef0` und `fDegree` hängt vom ausgewähltem `eKernelType` ab.

Maximale Iterationen

Für die Optimierung werden maximal so viele Iterationen genutzt, wie in `nMaxIterations` angegeben sind. Wenn der Wert `0` ist, wird der jeweilige Default-Wert verwendet.

Abbruchgrenze

Die Optimierung wird abgebrochen, sobald sich der Fehler zwischen zwei Iterationen nicht stärker verändert, als in `fEpsilon` angegeben ist. Wenn der Wert `0` ist, wird der jeweilige Default-Wert verwendet.

Anwendung

Ein SVM-Modell zur Klassifikation kann z.B. so erstellt werden:

```
hr := F_VN_CreateSvmModelExp(
    ipMlModel       := ipMlModel,
    eSvmType        := TCVN_SVM_NU_CLASSIFIER,
    fC              := 0,
    fNu             := 0.1,
    fP              := 0,
    eKernelType     := TCVN_SKT_RBF,
    fGamma          := 1,
    fCoef0          := 0,
    fDegree         := 0,
    nMaxIterations  := 0,
    fEpsilon        := 0,
    hrPrev          := hr);
```

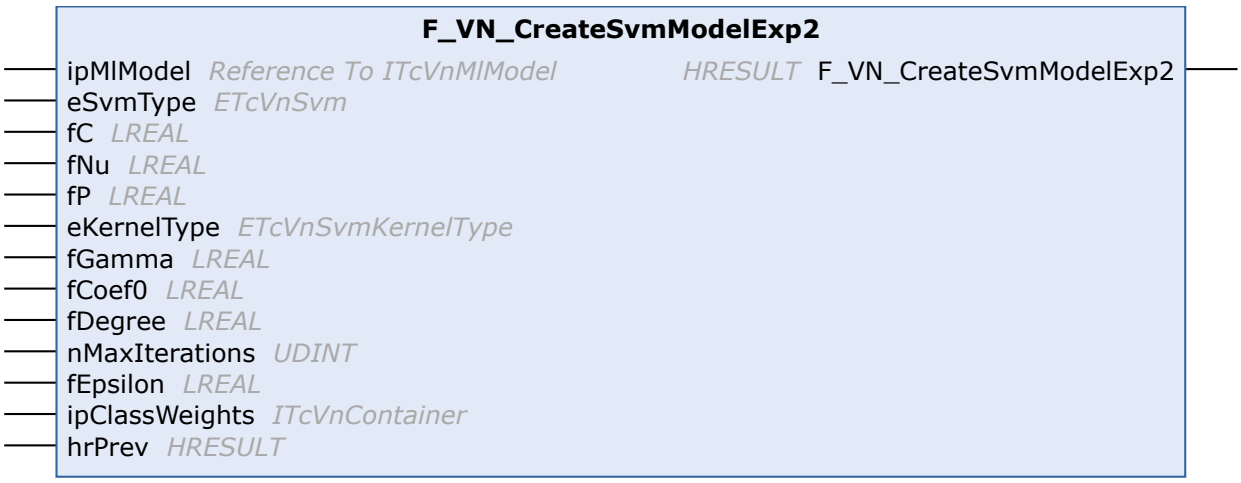
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.23 F_VN_CreateSvmModelExp2



Create an SVM model of the specified type. The initial reference count is set to one if a new model is created and kept, otherwise. Models of this type neither support on-line training (sample by sample) nor retraining. Predictions can only be scalar. (additional expert function for C support vector classifiers)

Syntax

Definition:

```
FUNCTION F_VN_CreateSvmModelExp2 : HRESULT
VAR_INPUT
    ipMlModel      : Reference To ITcVnMlModel;
    eSvmType       : ETcVnSvm;
    fC             : LREAL;
    fNu            : LREAL;
    fP             : LREAL;
    eKernelType    : ETcVnSvmKernelType;
    fGamma         : LREAL;
    fCoef0         : LREAL;
    fDegree        : LREAL;
    nMaxIterations : UDINT;
    fEpsilon       : LREAL;
    ipClassWeights : ITcVnContainer;
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created model (Non-zero interface pointers are reused.)
eSvmType	ETcVnSvm [▶ 230]	SVM model type
fC	LREAL	Parameter C (required for TCVN_SVM_C_CLASSIFIER, TCVN_SVM_EPS_REGRESSOR, and TCVN_SVM_NU_REGRESSOR; ignored otherwise)
fNu	LREAL	Parameter nu (required for TCVN_SVM_NU_CLASSIFIER, TCVN_SVM_NOVELTY_DETECTOR, and TCVN_SVM_NU_REGRESSOR; ignored otherwise)
fP	LREAL	Parameter p (required for TCVN_SVM_EPS_REGRESSOR; ignored otherwise)
eKernelType	ETcVnSvmKernelType [▶ 230]	Kernel type
fGamma	LREAL	Parameter gamma (used by polynomial, RBF, sigmoid, and chi-squared kernels; ignored otherwise)
fCoef0	LREAL	Parameter coef0 (used by polynomial and sigmoid kernels; ignored otherwise)
fDegree	LREAL	Degree (used by polynomial kernels; ignored otherwise)
nMaxIterations	UDINT	Maximum number of iterations (disabled if it equals 0 and fEpsilon is different from 0.0; triggers the usage of the default value of 100000 if nMaxIterations and fEpsilon equal 0)
fEpsilon	LREAL	Maximum allowed difference of the error between two successive iterations (disabled if it equals 0.0 and nMaxIterations is different from 0; triggers the usage of the default value of 0.00001 if nMaxIterations and fEpsilon equal 0)
ipClassWeights	ITcVnContainer [▶ 385]	Class weights (ContainerType_Vector_REAL or ContainerType_Vector_LREAL; only valid if eSvmType equals TCVN_SVM_C_CLASSIFIER; optional, set to 0 if not required or not allowed; default: 0)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreateSvmModelExp2` ist eine Expert-Variante von F_VN_CreateSvmModel [▶ 1449]. Sie enthält zusätzliche Parameter die nur für Modelle vom Typ `TCVN_SVM_C_CLASSIFIER` relevant sind.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Modell-Typ

`eSvmType` legt fest, ob das SVM-Modell zur Klassifizierung, zur Regression oder zur Anomalie-Detektion genutzt wird:

- TCVN_SVM_C_CLASSIFIER
- TCVN_SVM_NU_CLASSIFIER

- TCVN_SVM_NOVELTY_DETECTOR
- TCVN_SVM_EPS_REGRESSOR
- TCVN_SVM_NU_REGRESSOR

Modell-Parameter

Die Verwendung und Bedeutung der Parameter `fC`, `fNu` und `fP` hängt vom ausgewähltem `eSvmType` ab.

Kernel-Typ

Mit `eKernelType` wird der verwendete Kernel-Typ zur Berechnung festgelegt. Der Kernel-Typ hängt von der Aufgabenstellung / Datenverteilung ab und muss daran angepasst werden.

Kernel-Parameter

Die Verwendung und Bedeutung der Parameter `fGamma`, `fCoef0` und `fDegree` hängt vom ausgewähltem `eKernelType` ab.

Maximale Iterationen

Für die Optimierung werden maximal so viele Iterationen genutzt, wie in `nMaxIterations` angegeben sind. Wenn der Wert `0` ist, wird der jeweilige Default-Wert verwendet.

Abbruchgrenze

Die Optimierung wird abgebrochen, sobald sich der Fehler zwischen zwei Iterationen nicht stärker verändert, als in `fEpsilon` angegeben ist. Wenn der Wert `0` ist, wird der jeweilige Default-Wert verwendet.

Class-Weights

`ipClassWeights` ist ein Container, mit dem Wichtungen der einzelnen Klassen angegeben werden können.

Anwendung

Ein SVM-Modell zur Klassifikation parametrisiert mit Nu kann z.B. so erstellt werden:

```
hr := F_VN_CreateSvmModelExp2(
    ipMlModel       := ipMlModel,
    eSvmType        := TCVN_SVM_C_CLASSIFIER,
    fC              := 100,
    fNu             := 0,
    fP              := 0,
    eKernelType     := TCVN_SKT_RBF,
    fGamma          := 1,
    fCoef0          := 0,
    fDegree         := 0,
    nMaxIterations  := 0,
    fEpsilon        := 0,
    ipClassWeights  := ipClassWeights,
    hrPrev          := hr);
```

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.24 F_VN_CreateSvmSgdClassifier

Create a linear SVM classifier using stochastic gradient descent for training. The initial reference count is set to one if a new model is created and kept, otherwise. This SVM classifier is only applicable to binary classification problems. It learns a separating hyperplane between a class with label -1 and a class with label 1. These class labels are predefined. For training, any positive class labels are mapped to 1 and any negative class labels are mapped to -1. Models of this type neither support on-line training (sample by sample) nor retraining.

Syntax

Definition:

```
FUNCTION F_VN_CreateSvmSgdClassifier : HRESULT
VAR_INPUT
    ipMlModel : Reference To ITcVnMlModel;
    hrPrev    : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created model (Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreateSvmSgdClassifier` erstellt ein Support Vector Machine Modell mit linearem Kernel, das im Training Stochastic Gradient Descent (SVM-SGD) verwendet.

SVM-SGD-Modelle

Dieses SVM-Klassifikator Modell ist nur auf binäre Klassifikationsprobleme anwendbar. Es lernt eine trennende Hyperebene zwischen einer Klasse mit dem Label -1 und einer Klasse mit dem Label 1. Für das Training werden alle Samples gleichzeitig benötigt (Batch-Training) und ein Nachtrainieren ist nicht möglich.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Expert-Parameter

Die Expert-Variante F_VN_CreateSvmSgdClassifierExp [▶ 1459] enthält zusätzliche Parameter.

Anwendung

Ein SVM-SGD-Modell zur Klassifikation kann z.B. so erstellt werden:

```
hr := F_VN_CreateSvmSgdClassifier(
    ipMlModel   := ipMlModel,
    hrPrev      := hr);
```

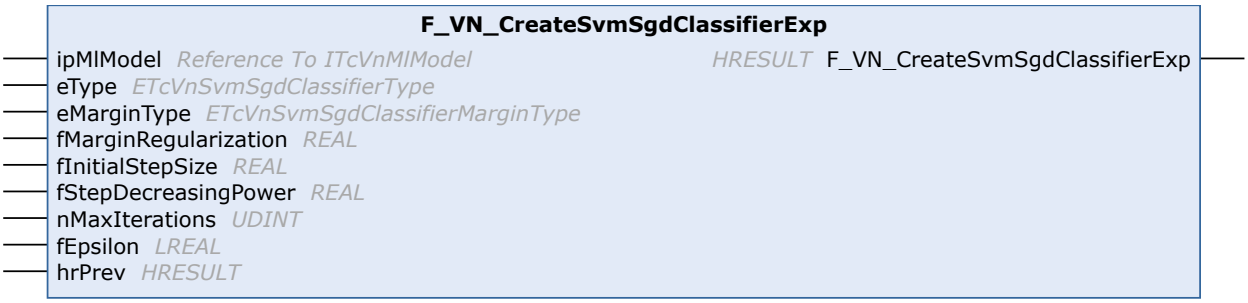
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.25 F_VN_CreateSvmSgdClassifierExp



Create a linear SVM classifier using stochastic gradient descent for training. The initial reference count is set to one if a new model is created and kept, otherwise. This SVM classifier is only applicable to binary classification problems. It learns a separating hyperplane between a class with label -1 and a class with label 1. These class labels are predefined. For training, any positive class labels are mapped to 1 and any negative class labels are mapped to -1. Models of this type neither support on-line training (sample by sample) nor retraining. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_CreateSvmSgdClassifierExp : HRESULT
VAR_INPUT
    ipMlModel             : Reference To ITcVnMlModel;
    eType                 : ETcVnSvmSgdClassifierType;
    eMarginType           : ETcVnSvmSgdClassifierMarginType;
    fMarginRegularization : REAL;
    fInitialStepSize      : REAL;
    fStepDecreasingPower  : REAL;
    nMaxIterations        : UDINT;
    fEpsilon              : LREAL;
    hrPrev                : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]	Returns the created model (Non-zero interface pointers are reused.)
eType	ETcVnSvmSgdClassifierType [▶ 231]	Learning algorithm type (default: TCVN_SSCT_ASGD)
eMarginType	ETcVnSvmSgdClassifierMarginType [▶ 231]	Margin type (default: TCVN_SSCMT_SOFT_MARGIN)
fMarginRegularization	REAL	Margin regularization parameter (default: 0.00001)
fInitialStepSize	REAL	Initial step size (default: 0.05)
fStepDecreasingPower	REAL	Power parameter (default: 0.75)
nMaxIterations	UDINT	Maximum number of iterations (disabled if it equals 0 and fEpsilon is different from 0.0; triggers the usage of the default value of 100000 if nMaxIterations and fEpsilon equal 0)
fEpsilon	LREAL	Maximum allowed difference of the error between two successive iterations (disabled if it equals 0.0 and nMaxIterations is different from 0; triggers the usage of the default value of 0.00001 if nMaxIterations and fEpsilon equal 0)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_CreateSvmSgdClassifierExp` ist eine Expert-Variante von F_VN_CreateSvmSgdClassifier [▶ 1458]. Sie enthält zusätzliche Parameter.

Parameter

Modell

In dem Interface Pointer `ipMlModel` wird das erstellte Modell zurückgegeben.

Algorithmus

`eType` legt fest welches Optimierungsverfahren im Training verwendet wird:

- TCVN_SSCT_SGD Stochastic Gradient Descent
- TCVN_SSCT_ASGD Average Stochastic Gradient Descent

Abgrenzung der Klassen

`eMarginType` legt fest, ob die Abgrenzung der Klassen strikt (`TCVN_SSCMT_HARD_MARGIN`) oder mit Ausreißern durchgeführt (`TCVN_SSCMT_SOFT_MARGIN`) werden soll.

Regularisierung

`fMarginRegularization` ist dafür verantwortlich, dass die Gewichte bei jedem Schritt abnehmen und steuert, wie stark der Einfluss von Ausreißern beschränkt wird. Je kleiner der Parameter, desto geringer die Wahrscheinlichkeit, dass ein Ausreißer ignoriert wird.

Schrittweite

`fInitialStepSize` gibt die Schrittweite zu Beginn des Trainings an.

Abnahme der Schrittweite

`fStepDecreasingPower` gibt den Exponenten für die Reduzierung der Schrittweite an.

Maximale Iterationen

Für die Optimierung werden maximal so viele Iterationen genutzt, wie in `nMaxIterations` angegeben sind. Wenn der Wert `0` ist, wird der jeweilige Default-Wert verwendet.

Abbruchgrenze

Die Optimierung wird abgebrochen, sobald sich der Fehler zwischen zwei Iterationen nicht stärker verändert, als in `fEpsilon` angegeben ist. Wenn der Wert `0` ist, wird der jeweilige Default-Wert verwendet.

Anwendung

Ein SVM-SGD-Modell zur Klassifikation kann z.B. so erstellt werden:

```
hr := F_VN_CreateSvmSgdClassifierExp(
    ipMlModel               := ipMlModel,
    eType                   := TCVN_SSCT_ASGD,
    eMarginType             := TCVN_SSCMT_SOFT_MARGIN,
    fMarginRegularization   := 0.00001,
    fInitialStepSize        := 0.05,
    fStepDecreasingPower    := 0.75,
    nMaxIterations          := 0,
    fEpsilon                := 0,
    hrPrev                  := hr);
```

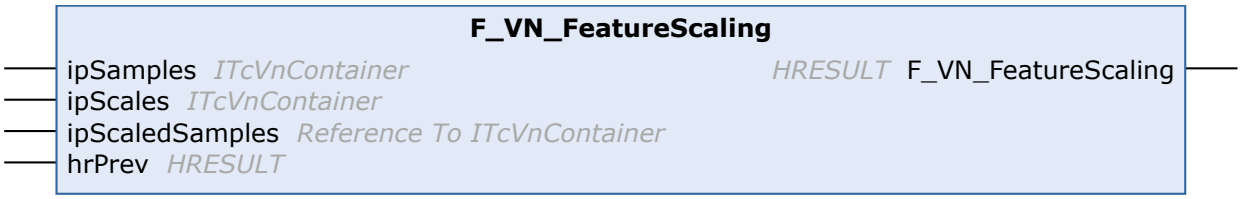
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.26 F_VN_FeatureScaling



Apply a feature scaling to one or more sample(s).

Syntax

Definition:

```
FUNCTION F_VN_FeatureScaling : HRESULT
VAR_INPUT
    ipSamples       : ITcVnContainer;
    ipScales        : ITcVnContainer;
    ipScaledSamples : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSamples	ITcVnContainer [▶ 385]	Container holding one or more input sample(s) (ContainerType_Vector_REAL, ContainerType_Vector_LREAL, ContainerType_Vector_Vector_REAL, or ContainerType_Vector_Vector_LREAL)
ipScales	ITcVnContainer [▶ 385]	Container with the scaling parameters of each feature (type ContainerType_Vector_REAL or ContainerType_Vector_LREAL).
ipScaledSamples	Reference To ITcVnContainer [▶ 385]	Returns the scaled sample(s) using the same type like ipSamples. If the same container ipSamples is uesd, the source data will be replaced.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_FeatureScaling` skaliert Merkmale von mehreren Samples. Für eine inverse Skalierung kann die Funktion F_VN_InverseFeatureScaling [▶ 1479] verwendet werden.

Parameter

Samples

Ein oder mehrere Samples müssen als Container an `ipSamples` übergeben werden.

Skalierung

Ein Container mit dem Skalierungstyp und den entsprechenden Skalierungsparametern muss an `ipScales` übergeben werden. Der Container kann durch die Funktion F_VN_GetFeatureScales [▶ 1473] berechnet oder nach dem dort beschriebenen Aufbau selbst erstellt werden.

Skalierte Samples

Die skalierten Samples werden über die Referenz `ipScaledSamples` zurückgegeben.

Expert-Parameter

Die Expert-Variante F_VN_FeatureScalingExp [▶ 1463] enthält zusätzliche Parameter.

Anwendung

Eine Merkmalsskalierung, bei der die Skalierungsparameter vorher durch F_VN_GetFeatureScales [▶ 1473] mit der `MINMAX`-Methode berechnet werden, sieht z.B. so aus:

```
hr := F_VN_GetFeatureScales(ipSamples, ipScales, TCVN_FST_MINMAX, hr);

hr := F_VN_FeatureScaling(
    ipSamples       := ipSamples,
    ipScales        := ipScales,
    ipScaledSamples := ipScaledSamples,
    hrPrev          := hr);
```

Verwandte Funktionen

- F_VN_GetFeatureScales [▶ 1473]
- F_VN_InverseFeatureScaling [▶ 1479]

- F_VN_InverseFeatureScaling_REAL [▶ 1481]

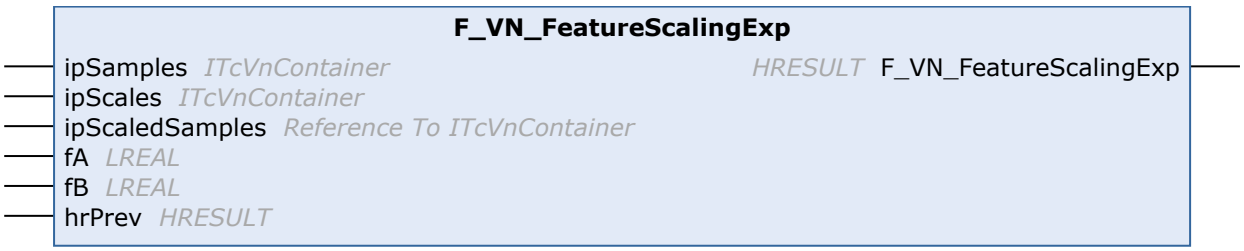
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.27 F_VN_FeatureScalingExp



Apply a feature scaling to one or more sample(s) and get output range between [fA, fB] in the case of TCVN_FST1_MINMAX. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_FeatureScalingExp : HRESULT
VAR_INPUT
    ipSamples       : ITcVnContainer;
    ipScales        : ITcVnContainer;
    ipScaledSamples : Reference To ITcVnContainer;
    fA              : LREAL;
    fB              : LREAL;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSamples	ITcVnContainer [▶ 385]	Container holding one or more input sample(s) (ContainerType_Vector_REAL, ContainerType_Vector_LREAL, ContainerType_Vector_Vector_REAL, or ContainerType_Vector_Vector_LREAL)
ipScales	ITcVnContainer [▶ 385]	Container with the scaling parameters of each feature (type ContainerType_Vector_REAL or ContainerType_Vector_LREAL).
ipScaledSamples	Reference To ITcVnContainer [▶ 385]	Returns the scaled sample(s) using the same type like ipSamples. If the same container ipSamples is uesd, the source data will be replaced.
fA	LREAL	represents the lower bound of the range for TCVN_FST1_MINMAX
fB	LREAL	represents the upper bound of the range for TCVN_FST1_MINMAX
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_FeatureScalingExp` ist die Expert-Variante von F_VN_FeatureScaling [▶ 1461]. Sie enthält zusätzliche Parameter.

Parameter

Samples

Ein oder mehrere Samples müssen als Container an `ipSamples` übergeben werden.

Skalierung

Ein Container mit dem Skalierungstyp und den entsprechenden Skalierungsparametern muss an `ipScales` übergeben werden. Der Container kann durch die Funktion F_VN_GetFeatureScales [▶ 1473] berechnet oder nach dem dort beschriebenen Aufbau selbst erstellt werden.

Skalierte Samples

Die skalierten Samples werden über die Referenz `ipScaledSamples` zurückgegeben.

Festlegen des Ausgabebereichs

Wenn der Skalierungstyp `TCVN_FST1_MINMAX` verwendet wird, kann über `fA` und `fB` zusätzlich der Wertebereich der Ausgabewerte vorgegeben werden.

Anwendung

Eine Merkmalsskalierung mit einem Ausgabewertebereich von 0,1 bis 0,9 sieht z.B. so aus:

```
hr := F_VN_FeatureScalingExp(
    ipSamples       := ipSamples,
    ipScales        := ipScales,
    ipScaledSamples := ipScaledSamples,
    fA              := 0.1,
    fB              := 0.9,
    hrPrev          := hr);
```

Verwandte Funktionen

- F_VN_GetFeatureScales [▶ 1473]
- F_VN_InverseFeatureScaling [▶ 1479]
- F_VN_InverseFeatureScaling_REAL [▶ 1481]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.28 F_VN_FeatureTransform

Apply a feature transform to one or more sample(s).

Syntax

Definition:

```
FUNCTION F_VN_FeatureTransform : HRESULT
VAR_INPUT
    ipFeatureTransform : ITcVnMlModel;
    ipSamples          : ITcUnknown;
    pipTransformed     : Pointer To ITcUnknown;
    hrPrev             : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipFeatureTransform	ITcVnMlModel [▶ 435]	Feature transform instance to be used
ipSamples	ITcUnknown [▶ 443]	Container holding one or more of input sample(s) (ContainerType_Vector_REAL, ContainerType_Vector_LREAL, ContainerType_Vector_Vector_REAL, or ContainerType_Vector_Vector_LREAL)
pipTransformed	Pointer To ITcUnknown [▶ 443]	Returns the transformed sample(s) using the same type like ipSamples
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_FeatureTransform` transformiert die Merkmale eines oder mehrerer Samples mit Hilfe eines Transformationsmodells. Eine inverse Transformation kann mit der Funktion F_VN_InverseFeatureTransform [▶ 1486] durchführt werden.

Parameter

Transformationsmodell

Zur Merkmalstransformation muss ein zuvor erstelltes Modell an `ipFeatureTransform` übergeben werden.

Samples

Ein Container mit den zu transformierenden Samples wird an `ipSamples` übergeben. Die folgenden Container-Typen sind erlaubt:

- `ContainerType_Vector_REAL` oder `ContainerType_Vector_LREAL` für ein einzelnes Sample
- `ContainerType_Vector_Vector_REAL` oder `ContainerType_Vector_Vector_LREAL` für einen Container mit mehreren Samples.

Transformierte Samples

Ein Container mit den transformieren Samples wird über den Pointer `pipTransformed` zurückgegeben. Der Container-Typ wird von `ipSamples` übernommen.

Anwendung

Eine Merkmalstransformation wird z.B. so durchgeführt:

```
hr := F_VN_FeatureTransform(
    ipFeatureTransform  := ipMlModel,
    ipSamples           := ipSamples,
    pipTransformed      := ADR(ipTransformed),
    hrPrev              := hr);
```

Verwandte Funktionen

- F_VN_InverseFeatureTransform [▶ 1486]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.29 F_VN_GetBatchClusters

Get the IDs of the best-matching clusters of a batch of samples.

Syntax

Definition:

```
FUNCTION F_VN_GetBatchClusters : HRESULT
VAR_INPUT
    ipClusterer : ITcVnMlModel;
    ipSamples   : ITcUnknown;
    ipClusters  : Reference To ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipClusterer	ITcVnMlModel [▶ 435]	Clusterer to be used
ipSamples	ITcUnknown [▶ 443]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipClusters	Reference To ITcVnContainer [▶ 385]	Returns the cluster IDs (ContainerType_Vector_DINT)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_GetBatchClusters` ermittelt für alle Samples in einem Container das jeweils am besten passende Cluster und gibt dessen ID zurück.

Parameter

Clusterer-Modell

Für die Zuordnung zu Clustern muss ein zuvor trainiertes Modell an `ipClusterer` übergeben werden.

Samples

Ein Container mit den Samples wird an `ipSamples` übergeben. Der Typ des Containers muss `ContainerType_Vector_Vector_REAL` oder `ContainerType_Vector_Vector_LREAL` sein.

Cluster-IDs

Ein Container mit den ermittelten IDs wird über die Referenz `ipClusters` zurückgegeben.

Expert-Parameter

Die Expert-Variante F_VN_GetBatchClustersExp [▶ 1467] enthält zusätzliche Parameter.

Anwendung

Die Zuordnung von Samples zu Clustern kann so berechnet werden:

```
hr := F_VN_GetBatchClusters(
    ipClusterer := ipClusterer,
    ipSamples   := ipSamples,
    ipClusters  := ipClusters,
    hrPrev      := hr);
```

Verwandte Funktionen

- F_VN_GetSampleCluster [▶ 1475]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.30 F_VN_GetBatchClustersExp

Get the IDs of the best-matching clusters of a batch of samples.

Syntax

Definition:

```
FUNCTION F_VN_GetBatchClustersExp : HRESULT
VAR_INPUT
    ipClusterer : ITcVnMlModel;
    ipSamples   : ITcUnknown;
    ipClusters  : Reference To ITcVnContainer;
    ipNovelties : Reference To ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipClusterer	ITcVnMlModel [▶ 435]	Clusterer to be used
ipSamples	ITcUnknown [▶ 443]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipClusters	Reference To ITcVnContainer [▶ 385]	Returns the cluster IDs (ContainerType_Vector_DINT)
ipNovelties	Reference To ITcVnContainer [▶ 385]	Returns the degree of novelty (0.0 if a sample is completely known; > 0.0 otherwise) of each sample (ContainerType_Vector_REAL; optional, set to 0 if not required)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_GetBatchClustersExp` ist die Expert-Variante von F_VN_GetBatchClusters [▶ 1466]. Sie enthält zusätzliche Parameter.

Parameter

Clusterer-Modell

Für die Zuordnung zu Clustern muss ein zuvor trainiertes Modell an `ipClusterer` übergeben werden.

Samples

Ein Container mit den Samples wird an `ipSamples` übergeben. Der Typ des Containers muss `ContainerType_Vector_Vector_REAL` oder `ContainerType_Vector_Vector_LREAL` sein.

Cluster-IDs

Ein Container mit den ermittelten IDs wird über die Referenz `ipClusters` zurückgegeben.

Anomalie-Stärken

Ein Container mit den Anomalie-Stärken jedes Samples wird über die Referenz `ipNovelties` zurückgegeben. Die Anomalie-Stärke gibt an, wie stark sich ein Sample von den Clustern unterscheidet. Eine Anomalie-Stärke von `0` gibt an, dass ein Sample exakt im Zentrum eines Clusters liegt.

Anwendung

Die Zuordnung von Samples zu Clustern inklusive der Anomalie-Stärken kann so berechnet werden:

```
hr := F_VN_GetBatchClustersExp(
    ipClusterer := ipClusterer,
    ipSamples   := ipSamples,
    ipClusters  := ipClusters,
    ipNovelties := ipNovelties,
    hrPrev      := hr);
```

Verwandte Funktionen

- F_VN_GetSampleCluster [▶ 1475]
- F_VN_GetSampleNovelty [▶ 1478]
- F_VN_GetBatchNovelties [▶ 1469]

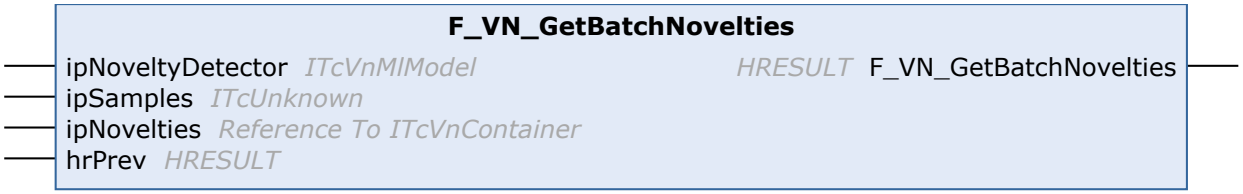
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.31 F_VN_GetBatchNovelties



Get the degree of novelty of multiple samples.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_GetBatchNovelties : HRESULT
VAR_INPUT
    ipNoveltyDetector : ITcVnMlModel;
    ipSamples         : ITcUnknown;
    ipNovelties       : Reference To ITcVnContainer;
    hrPrev            : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipNoveltyDetector	ITcVnMlModel [▶ 435]	Novelty detector to be used
ipSamples	ITcUnknown [▶ 443]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipNovelties	Reference To ITcVnContainer [▶ 385]	Returns the degree of novelty (0.0 if a sample is completely known; > 0.0 otherwise) of each sample (ContainerType_Vector_REAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_GetBatchNovelties` ermittelt für alle Samples in einem Container die Anomalie-Stärke.

Parameter

Anomalie-Modell

Für die Berechnung der Anomalie-Stärke muss das zuvor trainierte Modell an `ipNoveltyDetector` übergeben werden.

Samples

Ein Container mit den Samples wird an `ipSamples` übergeben.

Anomalie-Stärken

Ein Container mit den berechneten Anomalie-Stärken wird über die Referenz `ipNovelties` zurückgegeben.

Anwendung

Die Anomalie-Stärke von mehreren Samples kann so berechnet werden:

```
hr := F_VN_GetBatchNovelties(
    ipNoveltyDetector   := ipNoveltyDetector,
    ipSamples           := ipSamples,
    ipNovelties         := ipNovelties,
    hrPrev              := hr);
```

Verwandte Funktionen

- F_VN_GetSampleNovelty [▶ 1478]

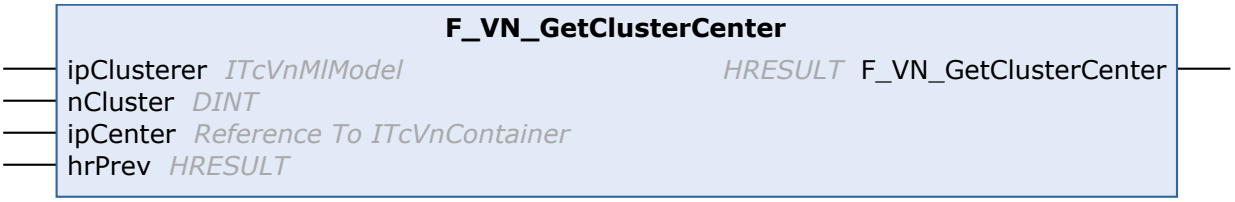
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.32 F_VN_GetClusterCenter



Get the center of a cluster.

Syntax

Definition:

```
FUNCTION F_VN_GetClusterCenter : HRESULT
VAR_INPUT
    ipClusterer : ITcVnMlModel;
    nCluster    : DINT;
    ipCenter    : Reference To ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipClusterer	ITcVnMlModel [▶ 435]	Clusterer to be used
nCluster	DINT	Cluster ID of the cluster the center of which is requested
ipCenter	Reference To ITcVnContainer [▶ 385]	Returns the cluster center (ContainerType_Vector_REAL or ContainerType_Vector_LREAL depending on the floating point type internally used by the clusterer)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_GetClusterCenter` gibt den Mittelpunkt eines Clusters zurück.

Parameter

Clusterer-Modell

Das Cluster, dessen Mittelpunkt berechnet werden soll, muss im zuvor trainierten Modell `ipClusterer` enthalten sein.

Cluster-ID

Die ID des entsprechenden Clusters wird an `nCluster` übergeben.

Cluster-Mittelpunkt

Der Mittelpunkt des Clusters wird über die Referenz `ipCenter` zurückgegeben. Der Typ des zurückgegebenen Containers ist `ContainerType_Vector_REAL` or `ContainerType_Vector_LREAL`, abhängig vom Datentyp, der intern in `ipClusterer` verwendet wird.

Anwendung

Der Mittelpunkt des dritten Clusters in einem Clusterer-Modell kann so ermittelt werden:

```
hr := F_VN_GetClusterCenter(
    ipClusterer := ipClusterer,
    nCluster    := 2,
    ipCenter    := ipCenter,
    hrPrev      := hr);
```

Verwandte Funktionen

- F_VN_GetClusterNum [▶ 1472]

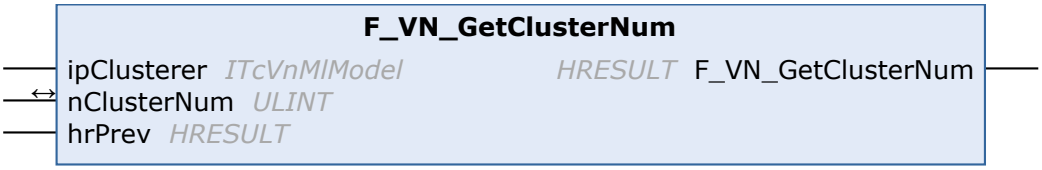
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.33 F_VN_GetClusterNum



Get the number of clusters used by this clusterer.

Syntax

Definition:

```
FUNCTION F_VN_GetClusterNum : HRESULT
VAR_INPUT
    ipClusterer : ITcVnMlModel;
END_VAR
VAR_IN_OUT
    nClusterNum : ULINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipClusterer	ITcVnMlModel [▶ 435]	Clusterer to be used
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nClusterNum	ULINT	Returns the number of clusters

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_GetClusterNum` gibt die Anzahl an Clustern in einem Clusterer-Modell zurück.

Parameter

Clusterer-Modell

Das zuvor trainierte Clusterer-Modell muss an `ipClusterer` übergeben werden.

Cluster-Anzahl

Die Anzahl der Cluster wird über `nClusterNum` zurückgegeben.

Anwendung

Die Anzahl der Cluster kann z.B. so ermittelt werden:

```
hr := F_VN_GetClusterNum(
    ipClusterer   := ipClusterer,
    nClusterNum   := nClusterNum,
    hrPrev        := hr);
```

Verwandte Funktionen

- F_VN_GetClusterCenter [▶ 1470]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.34 F_VN_GetFeatureScales

Calculate the scaling parameters for each feature in the input samples based on the scaling type.

Syntax

Definition:

```
FUNCTION F_VN_GetFeatureScales : HRESULT
VAR_INPUT
    ipSamples           : ITcVnContainer;
    ipScales            : Reference To ITcVnContainer;
    eFeatureScalingType : ETcVnFeatureScalingType;
    hrPrev              : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSamples	ITcVnContainer [▶ 385]	Container holding input sample(s) (ContainerType_Vector_REAL, ContainerType_Vector_LREAL, ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipScales	Reference To ITcVnContainer [▶ 385]	Returns a container with the scaling parameters for each feature (Vector_REAL or Vector_LREAL depending on the type of ipSamples).
eFeatureScalingType	ETcVnFeatureScalingType [▶ 213]	Feature scaling type
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_GetFeatureScales` berechnet Skalierungsparameter für jeden Merkmals-Wert der übergebenen Samples. Die entsprechende Skalierung kann im Anschluss mit der Funktion F_VN_FeatureScaling [▶ 1461] zur Merkmalsnormierung angewandt werden. Dieselbe Skalierung kann auch mit F_VN_InverseFeatureScaling [▶ 1479] zum Invertieren der Skalierung der Prädiktionsergebnisse einer Regression verwendet werden.

Parameter

Samples

Die Samples, auf deren Basis die Skalierungsparameter berechnet werden, werden als Container an `ipSamples` übergeben. Die Anzahl der Merkmals-Werte muss bei jedem Sample gleich sein.

Skalierung

Die berechneten Skalierungsparameter werden als Container über die Referenz `ipScales` zurückgegeben. Die Anzahl und Reihenfolge der Elemente des Containers hängen vom Skalierungstyp ab, der immer im ersten Element zu finden ist. Bei `MAXABS` folgt immer ein Skalierungswert pro Merkmal. Bei den anderen Typen sind es immer zwei Skalierungswerte pro Merkmal.

Skalierungstyp

Der gewünschte Skalierungstyp wird als Enum ETcVnFeatureScalingType [▶ 213] an `eFeatureScalingType` übergeben. Die Optionen sind:

- `TCVN_FST1_MAXABS`: Verwendet das Maximum der absoluten Merkmals-Werte, um eine Normierung im Bereich von -1 bis 1 zu erhalten. Wenn alle Werte positiv sind, erfolgt die Normierung von 0 bis 1.
- `TCVN_FST1_MINMAX`: Verwendet die Minimal- und Maximalwerte der Merkmals-Werte, um eine Normierung im Bereich von 0 bis 1 zu erhalten.
- `TCVN_FST1_STANDARDIZATION`: Verwendet den Mittelwert und die Standardabweichung der Merkmals-Werte, um eine Normierung der Vektor-Mittelwerte gleich 0 und der Vektor-Standardabweichung gleich 1 zu erhalten.

Anwendung

Die Berechnung von Skalierungsparametern nach der `MINMAX`-Methode sieht z.B. so aus:

```
hr := F_VN_GetFeatureScales(
    ipSamples           := ipSamples,
    ipScales            := ipScales,
    eFeatureScalingType := TCVN_FST1_MINMAX,
    hrPrev              := hr);
```

Wenn die Anzahl der Merkmals-Werte pro Sample in `ipSamples` z.B. `12` ist, wird `ipScales` für diesen Skalierungstyp exakt `1 + 2 * 12 = 25` Elemente haben.

Verwandte Funktionen

- F_VN_FeatureScaling [▶ 1461]
- F_VN_InverseFeatureScaling [▶ 1479]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.35 F_VN_GetSampleCluster

Get the ID of the best-matching cluster of a single sample.

Syntax

Definition:

```
FUNCTION F_VN_GetSampleCluster : HRESULT
VAR_INPUT
    ipClusterer : ITcVnMlModel;
    ipSample    : ITcUnknown;
END_VAR
VAR_IN_OUT
    nCluster    : DINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipClusterer	ITcVnMlModel [▶ 435]	Clusterer to be used
ipSample	ITcUnknown [▶ 443]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nCluster	DINT	Returns the ID of the cluster the sample has been assigned to

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_GetSampleCluster` ermittelt für ein Sample das am besten passende Cluster und gibt dessen ID zurück.

Parameter

Clusterer-Modell

Für die Zuordnung zu Clustern muss ein zuvor trainiertes Modell an `ipClusterer` übergeben werden.

Sample

Ein Container mit einem Sample wird an `ipSample` übergeben. Der Typ des Containers muss `ContainerType_Vector_REAL` oder `ContainerType_Vector_LREAL` sein.

Cluster-ID

Die ermittelte ID wird über `nCluster` zurückgegeben.

Expert-Parameter

Die Expert-Variante F_VN_GetSampleClusterExp [▶ 1476] enthält zusätzliche Parameter.

Anwendung

Die Zuordnung von einem Sample zu einem Cluster kann so berechnet werden:

```
hr := F_VN_GetSampleCluster(
    ipClusterer := ipClusterer,
    ipSample    := ipSample,
    nCluster    := nCluster,
    hrPrev      := hr);
```

Verwandte Funktionen

- F_VN_GetBatchClusters [▶ 1466]

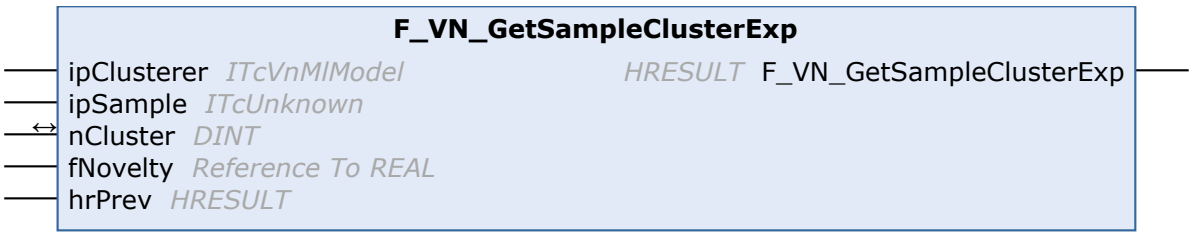
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.36 F_VN_GetSampleClusterExp



Get the ID of the best-matching cluster of a single sample. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_GetSampleClusterExp : HRESULT
VAR_INPUT
    ipClusterer : ITcVnMlModel;
    ipSample    : ITcUnknown;
END_VAR
VAR_IN_OUT
    nCluster    : DINT;
END_VAR
VAR_INPUT
    fNovelty    : Reference To REAL;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipClusterer	ITcVnMlModel [▶ 435]	Clusterer to be used
ipSample	ITcUnknown [▶ 443]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
fNovelty	Reference To REAL	Returns the degree of novelty (0.0 if a sample is completely known; > 0.0 otherwise) of the presented sample (optional, set to 0 if not required)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nCluster	DINT	Returns the ID of the cluster the sample has been assigned to

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_GetSampleClusterExp` ist die Expert-Variante von F_VN_GetSampleCluster [▶ 1475]. Sie enthält zusätzliche Parameter.

Parameter

Clusterer-Modell

Für die Zuordnung zu Clustern muss ein zuvor trainiertes Modell an `ipClusterer` übergeben werden.

Sample

Ein Container mit einem Sample wird an `ipSample` übergeben. Der Typ des Containers muss `ContainerType_Vector_REAL` oder `ContainerType_Vector_LREAL` sein.

Cluster-ID

Die ermittelte ID wird über `nCluster` zurückgegeben.

Anomalie-Stärke

Die Anomalie-Stärke des Samples wird über die Referenz `fNovelty` zurückgegeben. Die Anomalie-Stärke gibt an, wie stark sich ein Sample von den Clustern unterscheidet. Eine Anomalie-Stärke von 0 gibt an, dass ein Sample dem Modell bekannt ist.

Anwendung

Die Berechnung des Clusters und der Anomalie-Stärke eines Sample kann so berechnet werden:

```
hr := F_VN_GetSampleClusterExp(
    ipClusterer := ipClusterer,
    ipSample    := ipSample,
    nCluster    := nCluster,
    fNovelty    := fNovelty,
    hrPrev      := hr);
```

Verwandte Funktionen

- F_VN_GetSampleNovelty [▶ 1478]

- F_VN_GetBatchClusters [▶ 1466]
- F_VN_GetBatchNovelties [▶ 1469]

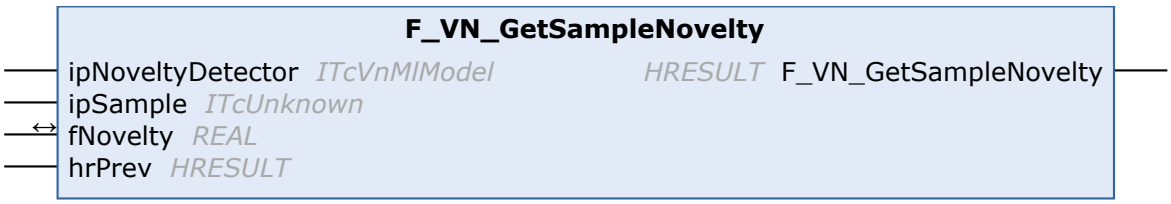
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.37 F_VN_GetSampleNovelty



Get the degree of novelty of a single sample.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_GetSampleNovelty : HRESULT
VAR_INPUT
    ipNoveltyDetector : ITcVnMlModel;
    ipSample          : ITcUnknown;
END_VAR
VAR_IN_OUT
    fNovelty          : REAL;
END_VAR
VAR_INPUT
    hrPrev            : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipNoveltyDetector	ITcVnMlModel [▶ 435]	Novelty detector to be used
ipSample	ITcUnknown [▶ 443]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fNovelty	REAL	Returns the degree of novelty (0.0 if a sample is completely known; > 0.0 otherwise) of the presented sample

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_GetSampleNovelty` ermittelt für ein Samples in einem Container die Anomalie-Stärke.

Parameter

Anomalie-Modell

Für die Berechnung der Anomalie-Stärke muss das zuvor trainierte Modell an `ipNoveltyDetector` übergeben werden.

Sample

Ein Container mit einem Sample wird an `ipSample` übergeben

Anomalie-Stärke

Die Anomalie-Stärke wird über die Referenz `fNovelty` zurückgegeben. Die Anomalie-Stärke gibt an, wie stark sich ein Sample von den gelernten Daten unterscheidet. Eine Anomalie-Stärke von 0 gibt an, dass ein Sample dem Modell bekannt ist.

Anwendung

Die Anomalie-Stärke eines Samples kann so berechnet werden:

```
hr := F_VN_GetSampleNovelty(
    ipNoveltyDetector   := ipNoveltyDetector,
    ipSample            := ipSample,
    fNovelty            := fNovelty,
    hrPrev              := hr);
```

Verwandte Funktionen

- F_VN_GetBatchNovelties [▶ 1469]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.38 F_VN_InverseFeatureScaling

Apply the inverse of feature scaling to one or more sample(s).

Syntax

Definition:

```
FUNCTION F_VN_InverseFeatureScaling : HRESULT
VAR_INPUT
    ipSamples        : ITcVnContainer;
    ipScales         : ITcVnContainer;
    ipInverseSamples : Reference To ITcVnContainer;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSamples	ITcVnContainer [▶ 385]	Container holding one or more input sample(s) (ContainerType_Vector_REAL, ContainerType_Vector_LREAL, ContainerType_Vector_Vector_REAL, or ContainerType_Vector_Vector_LREAL)
ipScales	ITcVnContainer [▶ 385]	Container with the scaling parameters of each feature (type ContainerType_Vector_REAL or ContainerType_Vector_LREAL).
ipInverseSamples	Reference To ITcVnContainer [▶ 385]	Returns the inversed sample(s) using the scaling type and parameters of ipScales. It has the same type of ipSamples. If the same container ipSamples is used, the source data will be replaced.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_InverseFeatureScaling` invertiert die Skalierung von einem oder mehreren Samples. Für eine Skalierung kann die Funktion F_VN_FeatureScaling [▶ 1461] verwendet werden.

Parameter

Samples

Ein oder mehrere Samples müssen als Container an `ipSamples` übergeben werden.

Skalierung

Ein Container mit dem Skalierungstyp und den entsprechenden Skalierungsparametern muss an `ipScales` übergeben werden. Der Container kann durch die Funktion F_VN_GetFeatureScales [▶ 1473] berechnet oder nach dem dort beschriebenen Aufbau selbst erstellt werden.

Skalierte Samples

Die invers skalierten Samples werden über die Referenz `ipInverseSamples` zurückgegeben.

Expert-Parameter

Die Expert-Variante F_VN_InverseFeatureScalingExp [▶ 1483] enthält zusätzliche Parameter.

Anwendung

Eine inverse Merkmalsskalierung, bei der die Skalierung vorher durch F_VN_FeatureScaling [▶ 1461] berechnet wurde, sieht z.B. so aus:

```
hr := F_VN_FeatureScaling(ipSamples, ipScales, ipSamplesScaled, hr);

hr := F_VN_InverseFeatureScaling(
    ipSamples       := ipSamplesScaled,
```

```
    ipScales        := ipScales,
    ipInverseSamples:= ipSamplesInvertedScaling,
    hrPrev          := hr);
```

Verwandte Funktionen

- F_VN_GetFeatureScales [▶ 1473]
- F_VN_FeatureScaling [▶ 1461]
- F_VN_InverseFeatureScaling_REAL [▶ 1481]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.39 F_VN_InverseFeatureScaling_REAL

Apply the inverse of feature scaling to a single value.

Syntax

Definition:

```
FUNCTION F_VN_InverseFeatureScaling_REAL : HRESULT
VAR_INPUT
    fSample         : REAL;
    ipScales        : ITcVnContainer;
END_VAR
VAR_IN_OUT
    fInverseSample : REAL;
END_VAR
VAR_INPUT
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fSample	REAL	Sample value
ipScales	ITcVnContainer [▶ 385]	Container with the scaling parameters of each feature (type ContainerType_Vector_REAL or ContainerType_Vector_LREAL).
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fInverseSample	REAL	Returns the inversed sample.

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_InverseFeatureScaling_REAL` invertiert die Skalierung von einem Sample mit nur einem Merkmal oder den Prädiktionswert eines skalaren Regressors. Für eine Skalierung kann die Funktion F_VN_FeatureScaling [▶ 1461] verwendet werden.

Parameter

Samples

Ein Sample vom Typ `REAL` muss an `fSample` übergeben werden.

Skalierung

Ein Container mit dem Skalierungstyp und den entsprechenden Skalierungsparametern muss an `ipScales` übergeben werden. Der Container kann durch die Funktion F_VN_GetFeatureScales [▶ 1473] berechnet oder nach dem dort beschriebenen Aufbau selbst erstellt werden.

Skaliertes Sample

Das invers skalierte Merkmal wird über `fInverseSample` zurückgegeben.

Expert-Parameter

Die Expert-Variante F_VN_InverseFeatureScalingExp_REAL [▶ 1484] enthält zusätzliche Parameter.

Anwendung

Eine inverse Merkmalsskalierung für ein Sample sieht z.B. so aus:

```
hr := F_VN_InverseFeatureScaling_REAL(
    fSample        := fSample,
    ipScales       := ipScales,
    fInverseSample := fSampleInvertedScaling,
    hrPrev         := hr);
```

Verwandte Funktionen

- F_VN_GetFeatureScales [▶ 1473]
- F_VN_FeatureScaling [▶ 1461]
- F_VN_InverseFeatureScaling [▶ 1479]

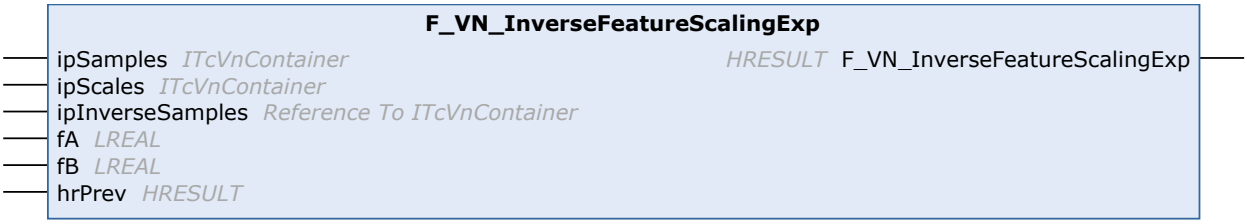
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.40 F_VN_InverseFeatureScalingExp



Apply the inverse of feature scaling to one or more sample(s). [fA, fB] should be used if the scaling has been performed using TCVN_FST1_MINMAX with a predetermined output range.. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_InverseFeatureScalingExp : HRESULT
VAR_INPUT
    ipSamples        : ITcVnContainer;
    ipScales         : ITcVnContainer;
    ipInverseSamples : Reference To ITcVnContainer;
    fA               : LREAL;
    fB               : LREAL;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSamples	ITcVnContainer [▶ 385]	Container holding one or more input sample(s) (ContainerType_Vector_REAL, ContainerType_Vector_LREAL, ContainerType_Vector_Vector_REAL, or ContainerType_Vector_Vector_LREAL)
ipScales	ITcVnContainer [▶ 385]	Container with the scaling parameters of each feature (type ContainerType_Vector_REAL or ContainerType_Vector_LREAL).
ipInverseSamples	Reference To ITcVnContainer [▶ 385]	Returns the inversed sample(s) using the scaling type and parameters of ipScales. It has the same type of ipSamples. If the same container ipSamples is used, the source data will be replaced.
fA	LREAL	represents the lower bound of the range for TCVN_FST1_MINMAX
fB	LREAL	represents the upper bound of the range for TCVN_FST1_MINMAX
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_InverseFeatureScalingExp` ist die Expert-Variante von F_VN_InverseFeatureScaling [▶ 1479]. Sie enthält zusätzliche Parameter.

Parameter

Samples

Ein oder mehrere Samples müssen als Container an `ipSamples` übergeben werden.

Skalierung

Ein Container mit dem Skalierungstyp und den entsprechenden Skalierungsparametern muss an `ipScales` übergeben werden. Der Container kann durch die Funktion F_VN_GetFeatureScales [▶ 1473] berechnet oder nach dem dort beschriebenen Aufbau selbst erstellt werden.

Skalierte Samples

Die invers skalierten Samples werden über die Referenz `ipInverseSamples` zurückgegeben.

Festlegen des Ausgabebereichs

Wenn der Skalierungstyp `TCVN_FST1_MINMAX` verwendet wird, kann über `fA` und `fB` zusätzlich der Wertebereich der Ausgabewerte vorgegeben werden.

Anwendung

Eine inverse Merkmalsskalierung, bei der die Skalierung vorher durch F_VN_FeatureScalingExp [▶ 1463] berechnet wurde, mit einem Ausgabewertebereich von 0,1 bis 0,9 sieht z.B. so aus:

```
hr := F_VN_FeatureScalingExp(ipSamples, ipScales, ipSamplesScaled, 0.1, 0.9, hr);

hr := F_VN_InverseFeatureScalingExp(
    ipSamples       := ipSamplesScaled,
    ipScales        := ipScales,
    ipInverseSamples:= ipSamplesInvertedScaling,
    fA              := 0.1,
    fB              := 0.9,
    hrPrev          := hr);
```

Verwandte Funktionen

- F_VN_GetFeatureScales [▶ 1473]
- F_VN_FeatureScaling [▶ 1461]
- F_VN_InverseFeatureScaling_REAL [▶ 1481]

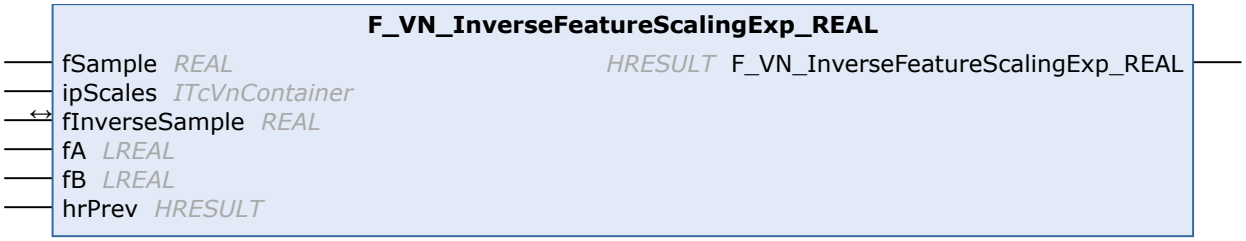
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.41 F_VN_InverseFeatureScalingExp_REAL



Apply the inverse of feature scaling to a single value. [fA, fB] should be used if the scaling has been performed using TCVN_FST1_MINMAX with a predetermined output range.

Syntax

Definition:

```
FUNCTION F_VN_InverseFeatureScalingExp_REAL : HRESULT
VAR_INPUT
    fSample         : REAL;
    ipScales        : ITcVnContainer;
END_VAR
VAR_IN_OUT
    fInverseSample : REAL;
END_VAR
VAR_INPUT
    fA              : LREAL;
    fB              : LREAL;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
fSample	REAL	Sample value
ipScales	ITcVnContainer [▶ 385]	Container with the scaling parameters of each feature (type ContainerType_Vector_REAL or ContainerType_Vector_LREAL).
fA	LREAL	represents the lower bound of the range for TCVN_FST1_MINMAX
fB	LREAL	represents the upper bound of the range for TCVN_FST1_MINMAX
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fInverseSample	REAL	Returns the inversed sample.

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_InverseFeatureScalingExp_REAL` ist die Expert-Variante von F_VN_InverseFeatureScaling_REAL [▶ 1481]. Sie enthält zusätzliche Parameter.

Parameter

Samples

Ein Sample vom Typ `REAL` muss an `fSample` übergeben werden.

Skalierung

Ein Container mit dem Skalierungstyp und den entsprechenden Skalierungsparametern muss an `ipScales` übergeben werden. Der Container kann durch die Funktion F_VN_GetFeatureScales [▶ 1473] berechnet oder nach dem dort beschriebenen Aufbau selbst erstellt werden.

Skaliertes Sample

Das invers skalierte Merkmal wird über `fInverseSample` zurückgegeben.

Festlegen des Ausgabebereichs

Wenn der Skalierungstyp `TCVN_FST1_MINMAX` verwendet wird, kann über `fA` und `fB` zusätzlich der Wertebereich der Ausgabewerte vorgegeben werden.

Anwendung

Eine inverse Merkmalsskalierung für ein Sample sieht z.B. so aus:

```
hr := F_VN_InverseFeatureScalingExp_REAL(
    fSample        := fSample,
    ipScales       := ipScales,
    fInverseSample := fSampleInvertedScaling,
    fA             := 0.1,
    fB             := 0.9,
    hrPrev         := hr);
```

Verwandte Funktionen

- F_VN_GetFeatureScales [▶ 1473]
- F_VN_FeatureScaling [▶ 1461]
- F_VN_InverseFeatureScaling_REAL [▶ 1481]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.42 F_VN_InverseFeatureTransform

Apply an inverse feature transform to one or more sample(s).

Syntax

Definition:

```
FUNCTION F_VN_InverseFeatureTransform : HRESULT
VAR_INPUT
    ipFeatureTransform : ITcVnMlModel;
    ipSamples          : ITcUnknown;
    pipTransformed     : Pointer To ITcUnknown;
    hrPrev             : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipFeatureTransform	ITcVnMlModel [▶ 435]	Feature transform instance to be used
ipSamples	ITcUnknown [▶ 443]	Container holding one or more of input sample(s) (ContainerType_Vector_REAL, ContainerType_Vector_LREAL, ContainerType_Vector_Vector_REAL, or ContainerType_Vector_Vector_LREAL)
pipTransformed	Pointer To ITcUnknown [▶ 443]	Returns the transformed sample(s) using the same type like ipSamples
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_InverseFeatureTransform` invertiert die Merkmalstransformation eines oder mehrerer Samples mit Hilfe eines Transformationsmodells. Die direkte Transformation kann mit dem jeweils selben Modell mit der Funktion F_VN_FeatureTransform [▶ 1465] durchführt werden.

Parameter

Transformationsmodell

Zur Merkmalstransformation muss ein zuvor erstelltes Modell an `ipFeatureTransform` übergeben werden.

Samples

Ein Container mit den zu transformierenden Samples wird an `ipSamples` übergeben. Die folgenden Container-Typen sind erlaubt:

- `ContainerType_Vector_REAL` oder `ContainerType_Vector_LREAL` für ein einzelnes Sample
- `ContainerType_Vector_Vector_REAL` oder `ContainerType_Vector_Vector_LREAL` für einen Container mit mehreren Samples.

Transformierte Samples

Ein Container mit den transformierten Samples wird über den Pointer `pipTransformed` zurückgegeben. Der Container-Typ wird von ipSamples übernommen.

Anwendung

Eine inverse Merkmalstransformation wird z.B. so durchgeführt:

```
hr := F_VN_InverseFeatureTransform(
    ipFeatureTransform  := ipMlModel,
    ipSamples           := ipSamplesTransformed,
    pipTransformed      := ADR(ipTransformed),
    hrPrev              := hr);
```

Verwandte Funktionen

- F_VN_FeatureTransform [▶ 1465]

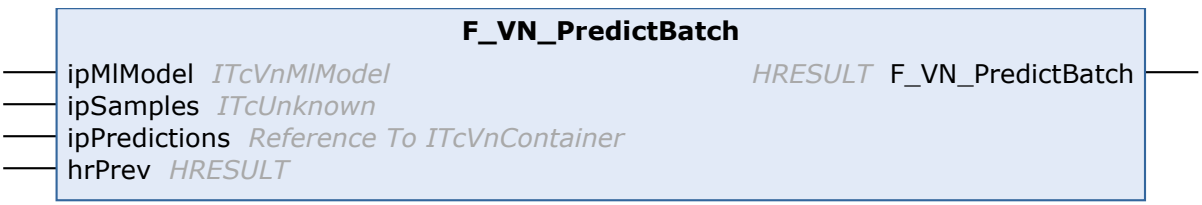
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.43 F_VN_PredictBatch



Compute predictions for a batch of samples.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_PredictBatch : HRESULT
VAR_INPUT
    ipMlModel     : ITcVnMlModel;
    ipSamples     : ITcUnknown;
    ipPredictions : Reference To ITcVnContainer;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	ITcVnMlModel [▶ 435]	Classifier or regressor to be used
ipSamples	ITcUnknown [▶ 443]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipPredictions	Reference To ITcVnContainer [▶ 385]	Returns the predicted outputs (depending on ipSamples; class labels (for classification, ContainerType_Vector_DINT) or real-valued predictions (for regression with scalar output, ContainerType_Vector_REAL or ContainerType_Vector_LREAL; for regression with vectorial output, ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL))
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_PredictBatch` berechnet die Prädiktion eines Modells für mehrere Samples. Abhängig von der Art des Models haben die Prädiktionen einen unterschiedlichen Datentyp (siehe `ipPredictions`).

Parameter

Modell

Zur Prädiktion muss das zuvor trainierte Modell an `ipMlModel` übergeben werden.

Samples

Die Samples werden in einem Container an `ipSamples` übergeben. Der Typ des Containers muss `ContainerType_Vector_Vector_REAL` oder `ContainerType_Vector_Vector_LREAL` sein.

Prädiktionen

Die Prädiktionen werden in einem Container über die Referenz `ipPredictions` zurückgegeben. Je nach Typ des Modells hat der Container einen unterschiedlichen Typ:

- Klassen-Label bei einer Klassifikation: `ContainerType_Vector_DINT`
- Numerische Prädiktionswerte bei einer skalaren Regression: `ContainerType_Vector_REAL` oder `ContainerType_Vector_LREAL`, abhängig von `ipSamples`

- Numerische Prädiktionswerte bei einer vektoriellen Regression: `ContainerType_Vector_Vector_REAL` oder `ContainerType_Vector_Vector_LREAL`, abhängig von `ipSamples`

Expert-Parameter

Die Expert-Variante F_VN_PredictBatchExp [▶ 1489] enthält zusätzliche Parameter.

Anwendung

Mehrere Samples können z.B. so gleichzeitig klassifiziert werden:

```
hr := F_VN_PredictBatch(
    ipMlModel       :=  ipMlModel,
    ipSamples       :=  ipSamples,
    ipPredictions   :=  ipPredictions,
    hrPrev          :=  hr);

hr := F_VN_GetAt_DINT(ipPredictions, nClassOfThirdSample, 2, hr);
```

Der Aufruf von `F_VN_GetAt_DINT` zeigt, wie aus den Prädiktionen das Klassenlabel des 3. Samples abgerufen werden kann.

Verwandte Funktionen

- F_VN_PredictSampleClass [▶ 1491]
- F_VN_PredictSampleScalar [▶ 1494]
- F_VN_PredictSampleVector [▶ 1497]

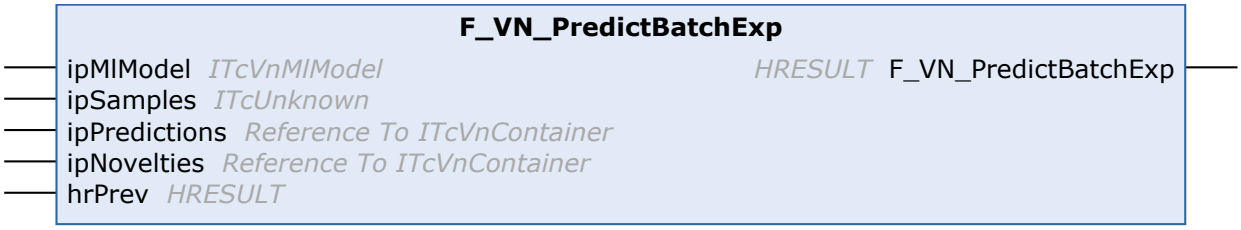
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.44 F_VN_PredictBatchExp



Compute predictions for a batch of samples. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_PredictBatchExp : HRESULT
VAR_INPUT
    ipMlModel     : ITcVnMlModel;
    ipSamples     : ITcUnknown;
    ipPredictions : Reference To ITcVnContainer;
    ipNovelties   : Reference To ITcVnContainer;
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	ITcVnMlModel [▶ 435]	Classifier or regressor to be used
ipSamples	ITcUnknown [▶ 443]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipPredictions	Reference To ITcVnContainer [▶ 385]	Returns the predicted outputs (depending on ipSamples; class labels (for classification, ContainerType_Vector_DINT) or real-valued predictions (for regression with scalar output, ContainerType_Vector_REAL or ContainerType_Vector_LREAL; for regression with vectorial output, ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL))
ipNovelties	Reference To ITcVnContainer [▶ 385]	Returns the degree of novelty (0.0 if a sample is completely known; > 0.0 otherwise) of each sample (ContainerType_Vector_REAL; optional, set to 0 if not required)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_PredictBatchExp` ist die Expert-Variante von F_VN_PredictBatch [▶ 1487]. Sie enthält zusätzlich Parameter.

Parameter

Modell

Zur Prädiktion muss das zuvor trainierte Modell an `ipMlModel` übergeben werden.

Samples

Die Samples werden in einem Container an `ipSamples` übergeben. Der Typ des Containers muss `ContainerType_Vector_Vector_REAL` oder `ContainerType_Vector_Vector_LREAL` sein.

Prädiktionen

Die Prädiktionen werden in einem Container über die Referenz `ipPredictions` zurückgegeben. Je nach Typ des Modells hat der Container einen unterschiedlichen Typ:

- Klassen-Label bei einer Klassifikation: `ContainerType_Vector_DINT`
- Numerische Prädiktionswerte bei einer skalaren Regression: `ContainerType_Vector_REAL` oder `ContainerType_Vector_LREAL`, abhängig von `ipSamples`
- Numerische Prädiktionswerte bei einer vektoriellen Regression: `ContainerType_Vector_Vector_REAL` oder `ContainerType_Vector_Vector_LREAL`, abhängig von `ipSamples`

Anomalie-Stärken

Die Anomalie-Stärken der Samples werden als Container über die Referenz `ipNovelties` zurückgegeben.

Anwendung

Mehrere Samples können z.B. so gleichzeitig klassifiziert werden:

```
hr := F_VN_PredictBatchExp(
    ipMlModel       := ipMlModel,
    ipSamples       := ipSamples,
    ipPredictions   := ipPredictions,
    ipNovelties     := ipNovelties,
    hrPrev          := hr);

hr := F_VN_GetAt_DINT(ipPredictions, nClassOfThirdSample, 2, hr);
```

Der Aufruf von `F_VN_GetAt_DINT` zeigt, wie aus den Prädiktionen das Klassenlabel des 3. Samples abgerufen werden kann.

Verwandte Funktionen

- F_VN_PredictSampleClass [▶ 1491]
- F_VN_PredictSampleScalar [▶ 1494]
- F_VN_PredictSampleVector [▶ 1497]
- F_VN_GetBatchNovelties [▶ 1478]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.45 F_VN_PredictSampleClass

Classify a single sample.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_PredictSampleClass : HRESULT
VAR_INPUT
    ipClassifier : ITcVnMlModel;
    ipSample     : ITcUnknown;
END_VAR
VAR_IN_OUT
    nClass       : DINT;
END_VAR
VAR_INPUT
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipClassifier	ITcVnMlModel [▶ 435]	Classifier to be used
ipSample	ITcUnknown [▶ 443]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nClass	DINT	Returns the classification result

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_PredictSampleClass` klassifiziert ein Sample basierend auf einem Klassifikationsmodell.

Parameter

Klassifikationsmodell

Das zuvor trainierte Klassifikationsmodell muss an `ipClassifier` übergeben werden.

Sample

Die Samples werden in einem Container an `ipSample` übergeben. Der Typ des Containers muss entweder `ContainerType_Vector_REAL` oder `ContainerType_Vector_LREAL` sein.

Klasse

Als Klassifikationsergebnis wird die Klasse des Sample über `nClass` zurückgegeben.

Expert-Parameter

Die Expert-Variante F_VN_PredictSampleClassExp [▶ 1493] enthält zusätzliche Parameter.

Anwendung

Ein Sample kann z.B. so klassifiziert werden:

```
hr := F_VN_PredictSampleClass(
    ipClassifier    := ipMlModel,
    ipSample        := ipSample,
    nClass          := nClassResult,
    hrPrev          := hr);
```

Verwandte Funktionen

- F_VN_PredictBatch [▶ 1487]

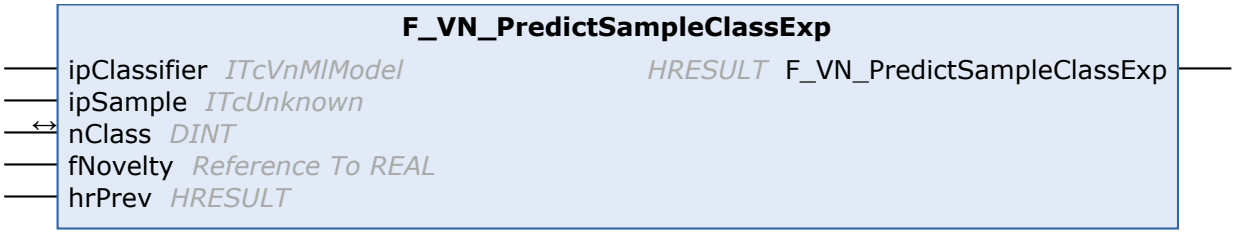
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.46 F_VN_PredictSampleClassExp



Classify a single sample. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_PredictSampleClassExp : HRESULT
VAR_INPUT
    ipClassifier : ITcVnMlModel;
    ipSample     : ITcUnknown;
END_VAR
VAR_IN_OUT
    nClass       : DINT;
END_VAR
VAR_INPUT
    fNovelty     : Reference To REAL;
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipClassifier	ITcVnMlModel [▶ 435]	Classifier to be used
ipSample	ITcUnknown [▶ 443]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
fNovelty	Reference To REAL	Returns the degree of novelty (0.0 if a sample is completely known; > 0.0 otherwise) of the presented sample (optional, set to 0 if not required)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nClass	DINT	Returns the classification result

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_PredictSampleClassExp` ist die Expert-Variante von F_VN_PredictSampleClass [▶ 1491]. Sie enthält zusätzliche Parameter.

Parameter

Klassifikationsmodell

Das zuvor trainierte Klassifikationsmodell muss an `ipClassifier` übergeben werden.

Sample

Die Samples werden in einem Container an `ipSample` übergeben. Der Typ des Containers muss entweder `ContainerType_Vector_REAL` oder `ContainerType_Vector_LREAL` sein.

Klasse

Als Klassifikationsergebnis wird die Klasse des Sample über `nClass` zurückgegeben.

Anomalie-Stärke

Die Anomalie-Stärke des Samples wird über `fNovelty` zurückgegeben.

Anwendung

Ein Sample kann z.B. so klassifiziert werden:

```
hr := F_VN_PredictSampleClassExp(
    ipClassifier    := ipMlModel,
    ipSample        := ipSample,
    nClass          := nClassResult,
    fNovelty        := fNovelty,
    hrPrev          := hr);
```

Verwandte Funktionen

- F_VN_GetSampleNovelty [▶ 1478]
- F_VN_PredictBatch [▶ 1487]

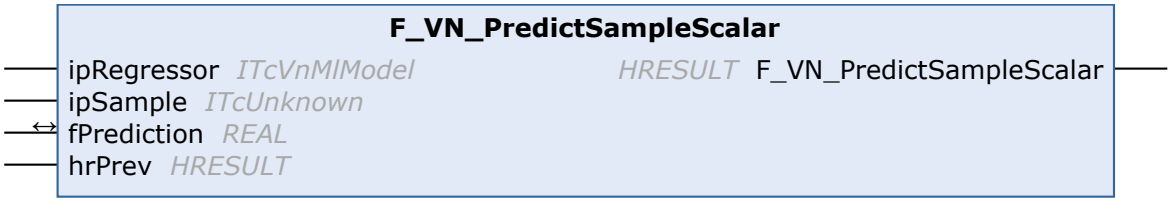
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.47 F_VN_PredictSampleScalar



Compute a scalar prediction for a single sample.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_PredictSampleScalar : HRESULT
VAR_INPUT
    ipRegressor : ITcVnMlModel;
    ipSample    : ITcUnknown;
END_VAR
VAR_IN_OUT
    fPrediction : REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipRegressor	ITcVnMlModel [▶ 435]	Regressor to be used
ipSample	ITcUnknown [▶ 443]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fPrediction	REAL	Returns the predicted output

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_PredictSampleScalar` berechnet den skalaren Prädiktionswert eines Samples basierend auf einem Regressionsmodell.

Parameter

Regressionsmodell

Das zuvor trainierte Regressionsmodell muss an `ipRegressor` übergeben werden.

Sample

Der Sample-Container wird als `ipSample` übergeben. Der Typ des Containers muss entweder `ContainerType_Vector_REAL` oder `ContainerType_Vector_LREAL` sein.

Prädiktion

Der berechnete Prädiktionswert wird über `fPrediction` zurückgegeben.

Expert-Parameter

Die Expert-Variante F_VN_PredictSampleScalarExp [▶ 1496] enthält zusätzliche Parameter.

Anwendung

Der Prädiktionswert eines Samples kann z.B. so berechnet werden:

```
hr := F_VN_PredictSampleScalar(
    ipRegressor := ipRegressor,
    ipSample    := ipSample,
    fPrediction := fPrediction,
    hrPrev      := hr);
```

Verwandte Funktionen

- F_VN_PredictBatch [▶ 1487]

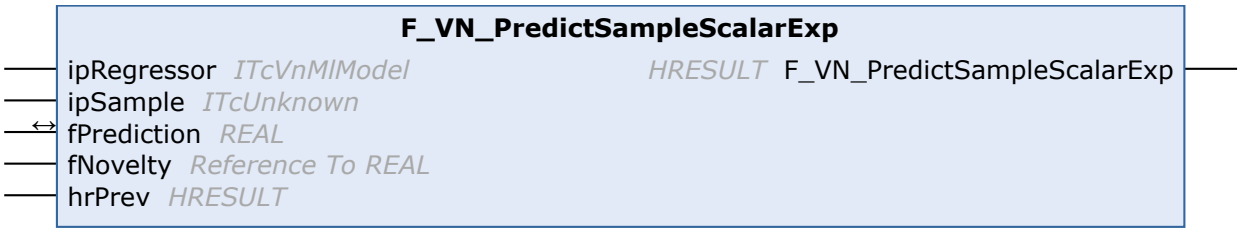
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.48 F_VN_PredictSampleScalarExp



Compute a scalar prediction for a single sample. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_PredictSampleScalarExp : HRESULT
VAR_INPUT
    ipRegressor : ITcVnMlModel;
    ipSample    : ITcUnknown;
END_VAR
VAR_IN_OUT
    fPrediction : REAL;
END_VAR
VAR_INPUT
    fNovelty    : Reference To REAL;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipRegressor	ITcVnMlModel [▶ 435]	Regressor to be used
ipSample	ITcUnknown [▶ 443]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
fNovelty	Reference To REAL	Returns the degree of novelty (0.0 if a sample is completely known; > 0.0 otherwise) of the presented sample (optional, set to 0 if not required)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fPrediction	REAL	Returns the predicted output

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_PredictSampleScalarExp` ist die Expert-Variante von F_VN_PredictSampleScalar [▶ 1494]. Sie enthält zusätzliche Parameter.

Parameter

Regressionsmodell

Das zuvor trainierte Regressionsmodell muss an `ipRegressor` übergeben werden.

Sample

Der Sample-Container wird als `ipSample` übergeben. Der Typ des Containers muss entweder `ContainerType_Vector_REAL` oder `ContainerType_Vector_LREAL` sein.

Prädiktion

Der berechnete Prädiktionswert wird über `fPrediction` zurückgegeben.

Anomalie-Stärke

Die Anomalie-Stärke des Samples wird über `fNovelty` zurückgegeben.

Anwendung

Der Regressionswert eines Samples kann z.B. so berechnet werden:

```
hr := F_VN_PredictSampleScalarExp(
    ipRegressor := ipRegressor,
    ipSample    := ipSample,
    fPrediction := fPrediction,
    fNovelty    := fNovelty,
    hrPrev      := hr);
```

Verwandte Funktionen

- F_VN_PredictBatch [▶ 1487]
- F_VN_GetBatchNovelties [▶ 1478]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.49 F_VN_PredictSampleVector

Compute a vectorial prediction for a single sample.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_PredictSampleVector : HRESULT
VAR_INPUT
    ipRegressor  : ITcVnMlModel;
    ipSample     : ITcUnknown;
    ipPrediction : Reference To ITcVnContainer;
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipRegressor	ITcVnMlModel [▶ 435]	Regressor to be used
ipSample	ITcUnknown [▶ 443]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
ipPrediction	Reference To ITcVnContainer [▶ 385]	Returns the predicted output (ContainerType_Vector_REAL or ContainerType_Vector_LREAL, depending on ipSample)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_PredictSampleVector` berechnet den Prädiktionsvektor eines Samples basierend auf einem Regressionsmodell.

Parameter

Regressionsmodell

Das zuvor trainierte Regressionsmodell muss an `ipRegressor` übergeben werden.

Sample

Der Sample-Container wird als `ipSample` übergeben. Der Typ des Containers muss entweder `ContainerType_Vector_REAL` oder `ContainerType_Vector_LREAL` sein.

Prädiktion

Der berechnete Prädiktionsvektor wird über die Referenz `ipPrediction` als Container zurückgegeben. Der Container-Typ wird von `ipSamples` übernommen.

Expert-Parameter

Die Expert-Variante F_VN_PredictSampleVectorExp [▶ 1499] enthält zusätzliche Parameter.

Anwendung

Der Prädiktionsvektor eines Samples kann z.B. so berechnet werden:

```
hr := F_VN_PredictSampleVector(
    ipRegressor := ipRegressor,
    ipSample    := ipSample,
    ipPrediction:= ipPrediction,
    hrPrev      := hr);
```

Verwandte Funktionen

- F_VN_PredictBatch [▶ 1487]

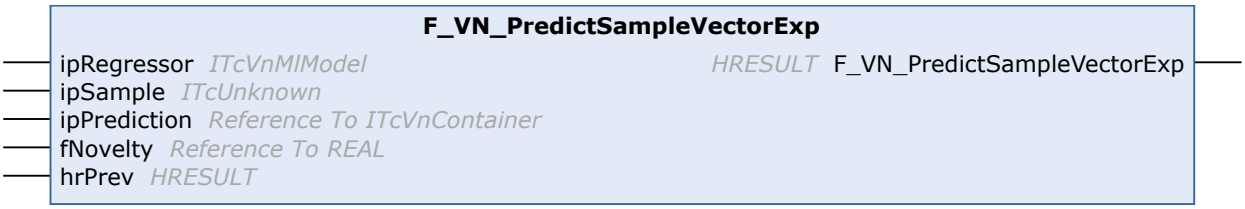
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.50 F_VN_PredictSampleVectorExp



Compute a vectorial prediction for a single sample. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_PredictSampleVectorExp : HRESULT
VAR_INPUT
    ipRegressor  : ITcVnMlModel;
    ipSample     : ITcUnknown;
    ipPrediction : Reference To ITcVnContainer;
    fNovelty     : Reference To REAL;
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipRegressor	ITcVnMlModel [▶ 435]	Regressor to be used
ipSample	ITcUnknown [▶ 443]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
ipPrediction	Reference To ITcVnContainer [▶ 385]	Returns the predicted output (ContainerType_Vector_REAL or ContainerType_Vector_LREAL, depending on ipSample)
fNovelty	Reference To REAL	Returns the degree of novelty (0.0 if a sample is completely known; > 0.0 otherwise) of the presented sample (optional, set to 0 if not required)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_PredictSampleVectorExp` ist die Expert-Variante von F_VN_PredictSampleVector [▶ 1497]. Sie enthält zusätzliche Parameter.

Parameter

Regressionsmodell

Das zuvor trainierte Regressionsmodell muss an `ipRegressor` übergeben werden.

Sample

Der Sample-Container wird als `ipSample` übergeben. Der Typ des Containers muss entweder `ContainerType_Vector_REAL` oder `ContainerType_Vector_LREAL` sein.

Prädiktion

Der berechnete Prädiktionsvektor wird über die Referenz `ipPrediction` als Container zurückgegeben. Der Container-Typ wird von `ipSamples` übernommen.

Anomalie-Stärke

Die Anomalie-Stärke des Sample wird über `fNovelty` zurückgegeben.

Anwendung

Der Prädiktionsvektor eines Samples kann z.B. so berechnet werden:

```
hr := F_VN_PredictSampleVectorExp(
    ipRegressor := ipRegressor,
    ipSample    := ipSample,
    ipPrediction:= ipPrediction,
    fNovelty    := fNovelty,
    hrPrev      := hr);
```

Verwandte Funktionen

- F_VN_PredictBatch [▶ 1487]
- F_VN_GetBatchNovelties [▶ 1478]

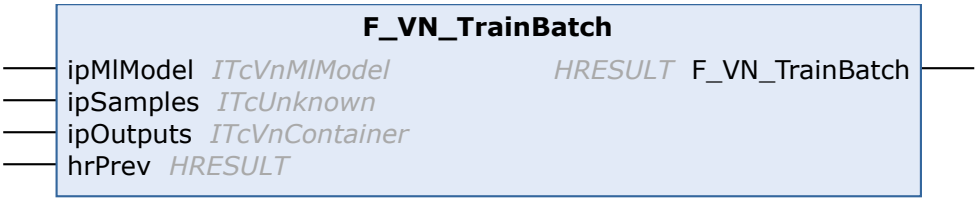
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.51 F_VN_TrainBatch



Train a classifier, a clusterer, a novelty detector, or a regressor with a batch of samples. On-line trainable models are trained once with each sample. Depending on the application and the number of available training samples, repeated training of such models with the same data may improve the results.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_TrainBatch : HRESULT
VAR_INPUT
    ipMlModel : ITcVnMlModel;
    ipSamples : ITcUnknown;
    ipOutputs : ITcVnContainer;
    hrPrev    : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	ITcVnMlModel [▶ 435]	Classifier, clusterer, novelty detector, or regressor to be used
ipSamples	ITcUnknown [▶ 443]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipOutputs	ITcVnContainer [▶ 385]	Class labels (for classification, ContainerType_Vector_DINT) or target outputs (for regression with scalar output, ContainerType_Vector_REAL or ContainerType_Vector_LREAL; for regression with vectorial output, ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL) to be learnt (must be 0 for clusterers and for novelty detectors)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_TrainBatch` trainiert ein Modell mit mehreren Samples. Abhängig von der Art des Modells müssen die Ergebniswerte einen anderen Typ haben (siehe `ipOutputs`). Als Modellart werden Klassifikator, Clusterer, Anomalie-Detektor und Regressor unterstützt.

Parameter

Modell

Das zuvor erstellte Modell muss an `ipMlModel` übergeben werden.

Samples

Die Samples werden in einem Container an `ipSamples` übergeben. Der Typ des Containers muss `ContainerType_Vector_Vector_REAL` oder `ContainerType_Vector_Vector_LREAL` sein.

Ergebnisse

Die bekannten Ergebnisse zu den Samples werden in einem Container an `ipOutputs` übergeben. Je nach Typ des Modells muss der Container einen unterschiedlichen Typ haben:

- Klassen-Label für eine Klassifikation: `ContainerType_Vector_DINT`
- Numerische Ergebniswerte für eine skalare Regression: `ContainerType_Vector_REAL` oder `ContainerType_Vector_LREAL`
- Numerische Ergebniswerte für eine vektorielle Regression: `ContainerType_Vector_Vector_REAL` oder `ContainerType_Vector_Vector_LREAL`

Wenn es sich um ein Clusterer-Modell oder um einen Anomalie-Detektor handelt, muss `ipOutputs := 0` sein.

Anwendung

Ein ML-Model zur Klassifikation kann z.B. so trainiert werden:

```
hr := F_VN_TrainBatch(
    ipMlModel   := ipMlModel,
    ipSamples   := ipSamples,
    ipOutputs   := ipClassLabels,
    hrPrev      := hr);
```

Verwandte Funktionen

- F_VN_TrainBatchClusters [▶ 1502]
- F_VN_TrainSample [▶ 1504]
- F_VN_TrainSampleClass [▶ 1505]
- F_VN_TrainSampleCluster [▶ 1507]
- F_VN_TrainSampleScalar [▶ 1508]
- F_VN_TrainSampleVector [▶ 1510]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.52 F_VN_TrainBatchClusters

Train a clusterer with a batch of samples and return the IDs of the clusters the samples have been assigned to, if requested. On-line trainable clusterers are trained once with each sample. Depending on the application and the number of available training samples, repeated training of such models with the same data may improve the results.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_TrainBatchClusters : HRESULT
VAR_INPUT
    ipClusterer : ITcVnMlModel;
    ipSamples   : ITcUnknown;
    ipClusters  : Reference To ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipClusterer	ITcVnMlModel [▶ 435]	Clusterer to be used
ipSamples	ITcUnknown [▶ 443]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipClusters	Reference To ITcVnContainer [▶ 385]	Returns the IDs of the clusters the samples have been assigned to (ContainerType_Vector_DINT; optional, set to 0 if not required)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_TrainBatchClusters` trainiert ein Clusterer-Modell mit mehreren Samples. Im Vergleich zu F_VN_TrainBatch [▶ 1500] können zusätzlich die zugewiesenen Cluster für jedes Sample zurückgegeben werden.

Parameter

Clusterer-Modell

Das zuvor erstelle Clusterer-Modell muss an `ipClusterer` übergeben werden.

Samples

Die Samples werden in einem Container an `ipSamples` übergeben. Der Typ des Containers muss `ContainerType_Vector_Vector_REAL` oder `ContainerType_Vector_Vector_LREAL` sein.

Clusters

Die IDs der zugewiesenen Cluster werden in einem Container vom Typ `ContainerType_Vector_DINT` über die Referenz `ipClusters` zurückgegeben.

Anwendung

Ein Clusterer-Modell kann z.B. so trainiert werden:

```
hr := F_VN_TrainBatchClusters(
    ipClusterer := ipClusterModel,
    ipSamples   := ipSamples,
    ipClusters  := ipClusters,
    hrPrev      := hr);

hr := F_VN_GetAt_DINT(ipClusters, nClusterOfThirdSample, 2, hr);
```

Der Aufruf der `F_VN_GetAt_DINT` zeigt, wie direkt nach dem Training die Cluster-Zuordnung des 3. Samples abgefragt werden kann.

Verwandte Funktionen

- F_VN_TrainBatch [▶ 1500]
- F_VN_TrainSample [▶ 1504]
- F_VN_TrainSampleClass [▶ 1505]
- F_VN_TrainSampleCluster [▶ 1507]
- F_VN_TrainSampleScalar [▶ 1508]

- F_VN_TrainSampleVector [▶ 1510]

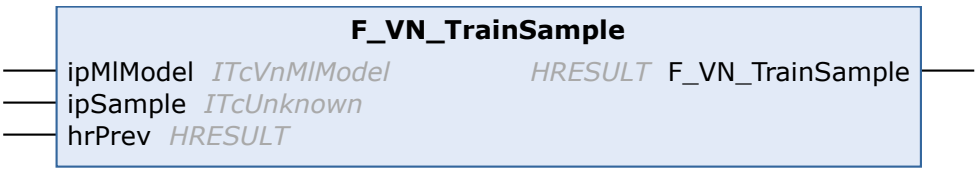
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.53 F_VN_TrainSample



Train a clusterer or a novelty detector with a single sample.

Syntax

Definition:

```
FUNCTION F_VN_TrainSample : HRESULT
VAR_INPUT
    ipMlModel : ITcVnMlModel;
    ipSample  : ITcUnknown;
    hrPrev    : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipMlModel	ITcVnMlModel [▶ 435]	Clusterer or novelty detector to be used
ipSample	ITcUnknown [▶ 443]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_TrainSample` trainiert ein Clusterer- oder Anomalie-Detektion-Modell mit einem Sample.

Parameter

Modell

Das zuvor erstellte Modell muss an `ipMlModel` übergeben werden.

Sample

Der Sample-Container wird an `ipSample` übergeben. Der Typ des Containers muss `ContainerType_Vector_REAL` oder `ContainerType_Vector_LREAL` sein.

Anwendung

Ein ML-Model kann mit einem Sample z.B. so trainiert werden:

```
hr := F_VN_TrainSample(
    ipMlModel   := ipMlModel,
    ipSample    := ipSample,
    hrPrev      := hr);
```

Verwandte Funktionen

- F_VN_TrainBatch [▶ 1500]
- F_VN_TrainBatchClusters [▶ 1502]
- F_VN_TrainSampleClass [▶ 1505]
- F_VN_TrainSampleCluster [▶ 1507]
- F_VN_TrainSampleScalar [▶ 1508]
- F_VN_TrainSampleVector [▶ 1510]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.54 F_VN_TrainSampleClass

Train a classifier with a single sample.

Syntax

Definition:

```
FUNCTION F_VN_TrainSampleClass : HRESULT
VAR_INPUT
    ipClassifier : ITcVnMlModel;
    ipSample     : ITcUnknown;
    nClass       : DINT;
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipClassifier	ITcVnMlModel [▶ 435]	Classifier to be used
ipSample	ITcUnknown [▶ 443]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
nClass	DINT	Class label to be learnt
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_TrainSampleClass` trainiert ein Klassifikator-Modell mit einem Sample.

Parameter

Klassifikator-Modell

Das zuvor erstellte Klassifikator-Modell muss an `ipClassifier` übergeben werden.

Sample

Der Sample-Container wird an `ipSample` übergeben. Der Typ des Containers muss `ContainerType_Vector_REAL` oder `ContainerType_Vector_LREAL` sein.

Klasse

Die Ziel-Klasse für das Sample wird als `nClass` übergeben.

Anwendung

Ein Klassifikator-Modell kann z.B. mit einzelnen Samples wie folgt trainiert werden:

```
hr := F_VN_TrainSampleClass(
    ipClassifier    := ipMlModel,
    ipSample        := ipSample,
    nClass          := nClass,
    hrPrev          := hr);
```

Verwandte Funktionen

- F_VN_TrainBatch [▶ 1500]
- F_VN_TrainBatchClusters [▶ 1502]
- F_VN_TrainSample [▶ 1504]
- F_VN_TrainSampleCluster [▶ 1507]
- F_VN_TrainSampleScalar [▶ 1508]
- F_VN_TrainSampleVector [▶ 1510]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.55 F_VN_TrainSampleCluster

Train a clusterer with a single sample and return the ID of the cluster the sample has been assigned to, if requested. The clusterer must be online trainable.

Syntax

Definition:

```
FUNCTION F_VN_TrainSampleCluster : HRESULT
VAR_INPUT
    ipClusterer : ITcVnMlModel;
    ipSample    : ITcUnknown;
    nCluster    : Reference To DINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipClusterer	ITcVnMlModel [▶ 435]	Clusterer to be used
ipSample	ITcUnknown [▶ 443]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
nCluster	Reference To DINT	Returns the ID of the cluster the sample has been assigned to (If the pointer is 0, no cluster ID is requested.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_TrainSampleCluster` trainiert ein Clusterer-Modell mit einem Sample und gibt die zugehörige Cluster ID des Samples zurück.

Parameter

Clusterer-Modell

Das zuvor erstellte Clusterer-Modell muss an `ipClusterer` übergeben werden.

Sample

Der Sample-Container wird als `ipSample` übergeben. Der Typ des Containers muss `ContainerType_Vector_REAL` oder `ContainerType_Vector_LREAL` sein.

Cluster

Die zum Sample zugehörige Cluster ID wird über `nCluster` zurückgegeben.

Anwendung

Ein Clusterer-Modell kann z.B. mit einem einzelnen Sample wie folgt trainiert werden:

```
hr := F_VN_TrainSampleCluster(
    ipClusterer     := ipClusterModel,
    ipSample        := ipSample,
    nCluster        := nCluster,
    hrPrev          := hr);
```

Verwandte Funktionen

- F_VN_TrainBatch [▶ 1500]
- F_VN_TrainBatchClusters [▶ 1502]
- F_VN_TrainSample [▶ 1504]
- F_VN_TrainSampleClass [▶ 1505]
- F_VN_TrainSampleScalar [▶ 1508]
- F_VN_TrainSampleVector [▶ 1510]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.56 F_VN_TrainSampleScalar

Train a regressor with a single sample and scalar output.

Syntax

Definition:

```
FUNCTION F_VN_TrainSampleScalar : HRESULT
VAR_INPUT
    ipRegressor : ITcVnMlModel;
    ipSample    : ITcUnknown;
    fOutput     : REAL;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipRegressor	ITcVnMlModel [▶ 435]	Regressor to be used
ipSample	ITcUnknown [▶ 443]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
fOutput	REAL	Scalar output to be learnt
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_TrainSampleScalar` trainiert ein skalares Regressor-Modell mit einem Sample.

Parameter

Regressor-Modell

Das zuvor erstellte Regressor-Modell muss an `ipRegressor` übergeben werden.

Sample

Der Sample-Container wird als `ipSample` übergeben. Der Typ des Containers muss `ContainerType_Vector_REAL` oder `ContainerType_Vector_LREAL` sein.

Trainingswert

Der Ergebniswert des Samples wird an `fOutput` übergeben.

Anwendung

Ein skalares Regressor-Modell kann z.B. wie folgt mit einem einzelnen Sample trainiert werden:

```
hr := F_VN_TrainSampleScalar(
    ipRegressor     := ipRegressorModel,
    ipSample        := ipSample,
    fOutput         := fOutput,
    hrPrev          := hr);
```

Verwandte Funktionen

- F_VN_TrainBatch [▶ 1500]
- F_VN_TrainBatchClusters [▶ 1502]
- F_VN_TrainSample [▶ 1504]
- F_VN_TrainSampleClass [▶ 1505]
- F_VN_TrainSampleCluster [▶ 1507]
- F_VN_TrainSampleVector [▶ 1510]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.19.57 F_VN_TrainSampleVector

Train a regressor with a single sample and vectorial output.

Syntax

Definition:

```
FUNCTION F_VN_TrainSampleVector : HRESULT
VAR_INPUT
    ipRegressor : ITcVnMlModel;
    ipSample    : ITcUnknown;
    ipOutput    : ITcVnContainer;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipRegressor	ITcVnMlModel [▶ 435]	Regressor to be used
ipSample	ITcUnknown [▶ 443]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
ipOutput	ITcVnContainer [▶ 385]	Vectorial output to be learnt (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_TrainSampleVector` trainiert ein vektorielles Regressor-Modell mit einem Sample.

Parameter

Regressor-Modell

Das zuvor erstelle Regressor-Modell muss an `ipRegressor` übergeben werden.

Sample

Der Sample-Container wird als `ipSample` übergeben. Der Typ des Containers muss `ContainerType_Vector_REAL` oder `ContainerType_Vector_LREAL` sein.

Trainingsvektor

Der Ergebnisvektor des Samples wird an `ipOutput` übergeben.

Anwendung

Ein vektorielles Regressor-Modell kann z.B. wie folgt mit einem einzelnen Sample trainiert werden:

```
hr := F_VN_TrainSampleVector(
    ipRegressor     := ipRegressorModel,
    ipSample        := ipSample,
    ipOutput        := ipVector,
    hrPrev          := hr);
```

Verwandte Funktionen

- F_VN_TrainBatch [▶ 1500]
- F_VN_TrainBatchClusters [▶ 1502]
- F_VN_TrainSample [▶ 1504]
- F_VN_TrainSampleClass [▶ 1505]
- F_VN_TrainSampleCluster [▶ 1507]
- F_VN_TrainSampleScalar [▶ 1508]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.20 Measurement

Die Gruppe enthält Funktionen zur Vermessung von Objekten auf Bildern.

Funktionen

Kantenlokalisierung

- LocateCircularArc [▶ 1520] zur Lokalisierung von anteiligen Kreisbögen
- LocateEdge [▶ 1526] zur Lokalisierung von linearen Kanten
- LocateEdges [▶ 1532] zur Lokalisierung von mehreren linearen Kanten
- LocateEllipse [▶ 1538] zur Lokalisierung von vollständigen Ellipsen

Geometrische Merkmale

- ClosestPointsBF [▶ 1518] zur Berechnung der nächstgelegenen Punkte zweier Punktewolken
- MeasureAngleBetweenEdges [▶ 1544] zur Vermessung des Winkels zwischen zwei Kanten
- MeasureEdgeDistance [▶ 1549] zur Vermessung des durchschnittlichen Abstands zweier Kanten
- MeasureMinEdgeDistance [▶ 1554] zur Vermessung des minimalen Abstands zweier Kanten

Sonstiges

- F_VN_AdjustSearchWindowOrientationToLinearEdge [▶ 1517] zur Ausrichtung des Suchfelds zu einer Kante

Parameter zur Kantenlokalisierung

Die Grenzen eines Bildobjekts liegen häufig nicht exakt auf der Grenze zwischen zwei Pixeln. Sie verlaufen stattdessen unabhängig von Pixelraster. Daher ist eine Subpixel-genaue Kantenlokalisierung nötig. Diese betrachtet nicht nur den Übergang zwischen zwei Pixeln, sondern bestimmt anhand des gesamten Intensitätsverlaufs einer Objektgrenze die Kantenposition innerhalb eines Pixels.

HINWEIS

Bild vor Vermessung nicht bearbeiten

Bearbeiten Sie das Bild vor dem Aufruf der Vermessungsfunktion nicht. Eine Bearbeitung kann zu Informationsverlust und entsprechend geringerer Genauigkeit führen.

Diese Grundfunktionalität der Kantenlokalisierung wird von allen Vermessungsfunktionen verwendet. Die Funktionen unterscheiden sich hauptsächlich darin, in welcher geometrischen Anordnung sie die Kantenlokalisierung durchführen (z. B. F_VN_LocateEllipse [▶ 1538] für runde Objekte) und welche Eigenschaften Sie daraus berechnen (z. B. F_VN_MeasureAngleBetweenEdges [▶ 1544] zur Winkelvermessung).

Bei jeder Funktion, die Kantenlokalisierung nutzt, müssen bestimmte Parameter definiert werden. Dazu gehören:

- Definition des Suchbereichs: Wo wird die Kante gesucht?
- Definition des Kantenkontrasts: Was wird als Kante erkannt?
- Definition des Vermessungsalgorithmus: Wie soll die genaue Position der Kante bestimmt werden.

Diese Parameter sind folgend beschrieben. Bei Expert-Parametern sind zudem Standard-Werte angegeben.

Suchbereich

Parameter	Beschreibung
Lage der Suchlinien	Die Lage der Suchlinien wird je nach zu lokalisierendem Merkmal definiert. Die Lage von Suchlinien für eine lineare Kante wird als einzelne Linie mit Startpunkt `aStartPoint` und Endpunkt `aEndPoint` definiert. Die Lage von Suchlinien für eine Ellipse wird durch einen Mittelpunkt `aCenter` und einen Radius `fSearchRadius` definiert. Die Lage von Suchlinien für Winkel wird durch einen Innenpunkt `aInnerPoint` und zwei Außenpunkte `aOuterPoint1` und `aOuterPoint2` definiert. Die Lage der Suchlinien sollte immer so eingestellt werden, dass die Suchlinien die zu lokalisierenden Kanten orthogonal schneiden. Ein rechteckiger Suchbereich kann bei Bedarf mit der Funktion F_VN_AdjustSearchWindowOrientationToLinearEdge [▶ 1517] orthogonal an eine Kante angepasst werden.
Anzahl der Suchlinien `nSearchLines : UDINT`	Hiermit wird festgelegt, wie viele Suchlinien für die Kantenlokalisierung verwendet werden sollen. Die einfachste Kantenlokalisierung hat eine Suchlinie, auf der die Kante gefunden werden muss. Bei mehreren Suchlinien werden je nach zu bestimmendem geometrischem Merkmal örtlich verteilt, sodass mehrere Kantenübergänge lokalisiert werden. Dies kann in folgenden Fällen nötig sein: • Nicht nur die Position, sondern auch der Verlauf oder die Rotation der Kante gefragt sind. • Das geometrische Merkmal (z. B. Ellipse) erfordert mehrere Suchlinien. • Ein Durchschnittswert ist gefordert. Bei der Lokalisierung von linearen Kanten werden die Suchlinien symmetrisch auf beiden Seiten der zentralen Suchlinie angeordnet, wobei die zentrale Suchlinie in `nSearchLines` inklusiv ist. Somit wird immer eine ungerade Anzahl an Suchlinien benötigt.
Abstand der Suchlinien `fSearchlineDist : REAL` *(Expert)* Standard: `1`	Die Suchlinien-Distanz definiert den Abstand zwischen benachbarten Suchlinien in Pixeln (> 0, default 1). Dadurch kann flexibel zwischen Genauigkeit und benötigter Rechenzeit abgewogen werden.
Invertierung der Suchrichtung `bInvertSearchDirection : BOOL` *(Expert)* Standard: `FALSE`	Der Lokalisierungsalgorithmus sucht die Kante immer von einem Ende der Suchlinie zum anderen. Z. B. wird von aStartPoint nach aEndPoint gesucht. Wenn es aufgrund der Bildbeschaffenheit nötig ist diese Suchrichtung zu invertieren (also vom anderen Ende der Suchlinien aus anzufangen), hilft dieser Parameter.

Kontrast der Kante

Parameter	Beschreibung
Kantenrichtung `eEdgeDirection : ETcVnEdgeDirection`	Die Kantenrichtung legt fest, ob ein Kantenübergang von Hell nach Dunkel oder von Dunkel nach Hell gefunden werden soll. Dazu stehen folgende Werte zur Verfügung: • `TCVN_ED_DARK_TO_LIGHT` • `TCVN_ED_LIGHT_TO_DARK`
Minimalstärke `fMinStrength : LREAL`	Die Minimalstärke legt fest, wie groß der Intensitätsunterschied innerhalb einer Kante mindestens sein muss, damit sie gefunden wird. Dadurch lässt sich steuern, wie viel Kontrast eine Kante haben muss, um gefunden zu werden.
Maximaldicke `nMaxThickness : UDINT` *(Expert)* Standard: `10`	Die Maximaldicke legt fest, innerhalb von wie vielen Pixeln `fMinStrength` erreicht werden muss. Dadurch lässt sich steuern, ob nur scharfe Kanten oder auch sehr unscharfe Kanten gefunden werden sollen.

Algorithmus

Parameter	Beschreibung
Algorithmus `eAlgorithm : ETcVnEdgeDetectionAlgorithm` *(Expert)* Standard: `TCVN_EDA_INTERPOLATION`	Dieser Parameter definiert den verwendeten Lokalisierungsalgorithmus. Zur Verfügung stehen: • `TCVN_EDA_INTERPOLATION` • `TCVN_EDA_APPROX_ERF` • `TCVN_EDA_APPROX_GAUSSIAN` `TCVN_EDA_APPROX_ERF` und `TCVN_EDA_APPROX_GAUSSIAN` sind langsamer als der Standard-Algorithmus `TCVN_EDA_INTERPOLATION`, können aber eine höhere Genauigkeit erreichen. Voraussetzung dafür ist allerdings, dass der Intensitätsverlauf der Kante auch möglichst gut dem vorgegebenen Modell entspricht. Als Modell werden die Fehlerfunktion `erf` bzw. die Gauß-Funktion verwendet. Die `erf`-Funktion ist gut für Kanten geeignet, wenn sich der Übergang von hell nach dunkel über mehrere Pixel erstreckt und auf beiden Seiten der helle bzw. dunkle Bereich in etwa konstant bleibt. Eine Gauß-Funktion ist gut für Linien geeignet, d. h. ein Übergang dunkel-hell-dunkel oder umgekehrt. Das Helligkeitsplateau darf dabei aber nur wenige Pixel breit sein, ansonsten sollte die erf-Funktion verwendet werden.
Subpixel / Iterationen `nSubpixelsIterations : UDINT` *(Expert)* Standard: 10	Die Bedeutung dieses Parameters ist abhängig vom verwendeten Lokalisierungsalgorithmus. Bei `TCVN_EDA_INTERPOLATION`: Die Anzahl der betrachteten Subpixel definiert, wie feingranular die Suchlinie die unterliegenden Pixelintensitäten auswertet. Ein Wert von 10 gibt z. B. an, dass auf der Länge von 1 Pixel 10 Messwerte genommen werden. Empfohlen: `5` - `10` Bei `TCVN_EDA_APPROX_ERF` und `TCVN_EDA_APPROX_GAUSSIAN`: Die Anzahl der Iterationen ist ein Abbruchkriterium, das definiert, mit maximal wie vielen Iterationen das Kantenmodell auf die Pixelintensitäten approximiert werden sollen. Mehr Iterationen erhöhen die Laufzeit, wenn diese zu hoch ist sollte die Interpolationsmethode verwendet werden. Empfohlen: `50` - `100`
Näherungsgenauigkeit `fApproxPrecision : REAL` *(Expert)* Standard: `0.001`	Die Näherungsgenauigkeit ist ein weiteres Abbruchkriterium der Kantenlokalisierung. Sie wird nur beachtet, wenn der Lokalisierungsalgorithmus `TCVN_EDA_APPROX_ERF` oder `TCVN_EDA_APPROX_GAUSSIAN` ist.

Parameter	Beschreibung
	Die Approximation der Kante bzw. der Linie wird abgebrochen, wenn entweder die Anzahl der Iterationen von `nSubPixelIterations` erreicht oder die relative Abweichung in Pixeln zwei aufeinanderfolgender Iterationen kleiner als `fApproximationPrecision` ist Empfohlen: `0.01` - `0.0001`

Interpretation des HRESULT

Für Measurement-Funktionen haben die HRESULT [▶ 150]–Rückgabewerte folgende Bedeutung:

Code	Name	Beschreibung
16#000	S_OK	Funktion wurde erfolgreich ausgeführt und auf allen Suchlinien wurde eine Kante lokalisiert.
16#001	S_FALSE	Funktion wurde erfolgreich ausgeführt, aber es wurde nicht auf jeder Suchlinie eine Kante lokalisiert.
16#256	S_WATCHDOGTIMEOUT	Funktion wurde vom Watchdog abgebrochen. In diesem Fall können alle Measurement-Funktionen Teilergebnisse zurückgeben.
16#70C	NOTFOUND	Funktion wurde erfolgreich ausgeführt, aber auf keiner Suchlinie konnte eine Kante gefunden werden. Es wird davon ausgegangen, dass eine Vermessungsfunktion nur aufgerufen wird, wenn sichergestellt ist, dass das Bild ein Objekt mit den parametrierten Kanteneigenschaften enthält. Ein Nicht-Finden aller Suchlinien deutet dabei auf eine falsche Parametrierung oder einen ungeeigneten Aufruf der Funktion hin. Daher wird in diesem Fall ein Error-Code ausgegeben.
16#7xx	Alle Error-Codes außer NOTFOUND	Funktion wurde nicht erfolgreich ausgeführt. Siehe dazu: ADS Return Codes [▶ 2966]

Zur Verarbeitung des Rückgabewerts kann beispielhaft folgende Abfrage benutzt werden:

```
IF SUCCEEDED(hr) THEN
    // Process results
    IF hr <> S_OK THEN
        // Emit warning that not all search lines were found
    END_IF
ELSIF hr = Tc2_System.E_HRESULTAdsErr.NOTFOUND THEN
    // Emit warning for wrong parametrization or missing piece
ELSE
    // Error handling
END_IF
```

Ausführungszeit reduzieren

Die Ausführungszeit schwankt bei einem festen Suchfenster leicht je nach Position des Bauteils im Bild. Ist die gefundene Kante näher am Startpunkt, ist die Ausführungszeit kürzer als bei einer weiter entfernten Kante.

Ein größeres Schwanken kann bei den beiden Approximierungs-Algorithmen auftreten, da diese unterschiedlich viele Iterationen benötigen, um die Modellparameter mittels der umliegenden Pixelintensitäten zu approximieren.

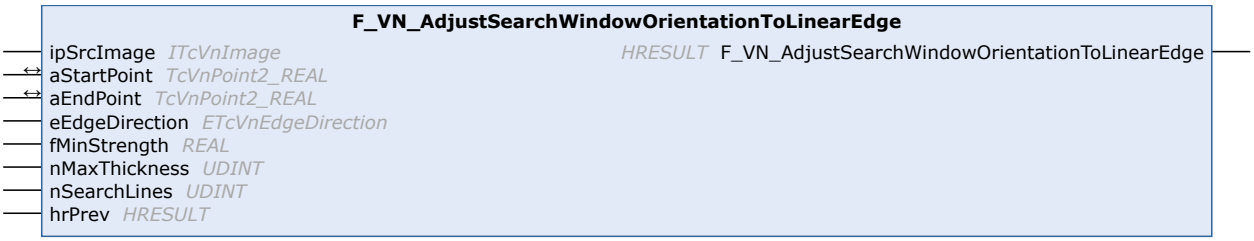
Generell werden weniger Iterationen benötigt, wenn die Suchlinien des Suchfensters möglichst orthogonal auf die Kante treffen. Falls die Position und Orientierung der Objekte im Bild keinen oder nur geringen Schwankungen unterliegen, richten Sie das Suchfenster entsprechend aus, um kürzere Ausführungszeiten zu erreichen. Zudem lässt sich die maximal benötigte Zeit reduzieren, indem die maximale Anzahl an Iterationen verringert wird, was aber ggf. auch ungenauere Ergebnisse liefert.

Zudem kann die Zeit auch mit einem Watchdog von außen begrenzt werden, der die Ausführung der Funktion bei Bedarf abbricht und die vorhandenen Teilergebnisse zurückliefert.

Beispiele

Measurement [▶ 2900]

6.1.4.20.1 F_VN_AdjustSearchWindowOrientationToLinearEdge



Adjust the search window to be orthogonal to a linear edge. Might be helpful before calling edge localization and distance measurement functions that rely on the search window.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_AdjustSearchWindowOrientationToLinearEdge : HRESULT
VAR_INPUT
    ipSrcImage     : ITcVnImage;
END_VAR
VAR_IN_OUT
    aStartPoint    : TcVnPoint2_REAL;
    aEndPoint      : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    eEdgeDirection : ETcVnEdgeDirection;
    fMinStrength   : REAL;
    nMaxThickness  : UDINT;
    nSearchLines   : UDINT;
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
eEdgeDirection	ETcVnEdgeDirection [▶ 210]	Specification of the edge direction to search for
fMinStrength	REAL	Specification of the minimum strength (intensity difference) of the edge to search for
nMaxThickness	UDINT	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels
nSearchLines	UDINT	Width of the search window, i.e. the number of search lines (3, 5, 7, ...), centered around the line specified by aStartPoint and aEndPoint
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aStartPoint	TcVnPoint2_REAL [▶ 171]	Position from which to start the search process (in the direction of aEndPoint), which is not changed
aEndPoint	TcVnPoint2_REAL [▶ 171]	Position where the search process ends, which is adjusted by this function

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_AdjustSearchWindowOrientationToLinearEdge` rotiert das Suchfeld für Vermessungsfunktionen derart, dass es orthogonal zu einer linearen Kante liegt.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss eine Kante entsprechend den Parametern zur Kantendetektion enthalten.

Suchfeld (Rückgabewert)

Das anzupassende Suchfeld wird als Referenz übergeben, sodass es angepasst werden kann. Das Suchfeld besteht aus der zentralen Suchlinie bestehend aus Startpunkt `aStartPoint` und Endpunkt `aEndPoint` sowie der Anzahl der Suchlinien `nSearchLines`.

Nach dem Funktionsaufruf enthalten `aStartPoint` und `aEndPoint` die angepasste, zentrale Suchlinie. Diese ist nun orthogonal zur linearen Kante auf dem Eingangsbild `ipSrcImage`.

Parameter zur Kantenlokalisierung

Die Parameter `eEdgeDirection`, `fMinStrength` und `nMaxThickness` bestimmen die Suchmerkmale für die lineare Kante auf dem Eingangsbild `ipSrcImage`. Diese funktionieren analog zur Kantenlokalisierung in F_VN_LocateEdge [▶ 1526].

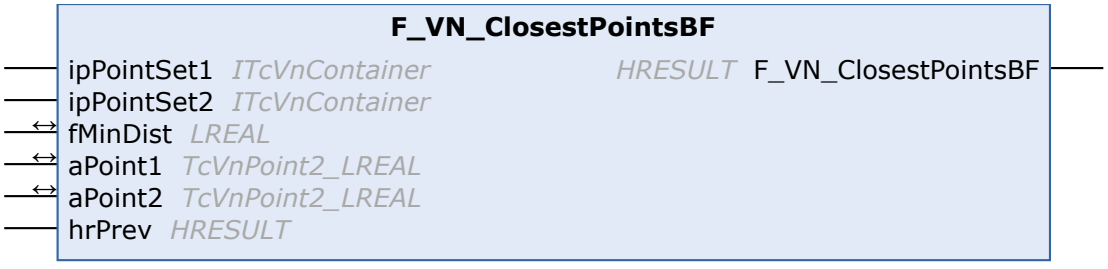
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.20.2 F_VN_ClosestPointsBF



Find the closest distance between two 2d point sets of the same type, using a brute force approach.

Syntax

Definition:

```
FUNCTION F_VN_ClosestPointsBF : HRESULT
VAR_INPUT
    ipPointSet1 : ITcVnContainer;
    ipPointSet2 : ITcVnContainer;
END_VAR
VAR_IN_OUT
```

```
    fMinDist    : LREAL;
    aPoint1     : TcVnPoint2_LREAL;
    aPoint2     : TcVnPoint2_LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipPointSet1	ITcVnContainer [▶ 385]	Container with 1st set of 2d points
ipPointSet2	ITcVnContainer [▶ 385]	Container with 2nd set of 2d points
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fMinDist	LREAL	Returns the minimum distance between 2 points of the different point sets
aPoint1	TcVnPoint2_LREAL [▶ 171]	Returns the point out of ipPointSet1, for which fMinDist is achieved
aPoint2	TcVnPoint2_LREAL [▶ 171]	Returns the point out of ipPointSet2, for which fMinDist is achieved

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_ClosestPointsBF` bestimmt, welche Punkte aus zwei Punktewolken den geringsten Abstand haben.

Parameter

Punktewolken

In den Punktewolken `ipPointSet1` und `ipPointSet2` wird nach den nächstgelegenen beiden Punkten gesucht.

Minimale Distanz (Rückgabewert)

Der Parameter `fMinDist` gibt die Distanz der beiden nächstgelegenen Punkte zurück. Dies entspricht der minimalen Distanz der beiden Punktewolken.

Nächstgelegene Punkte (Rückgabewert)

Die Parameter `aPoint1` und `aPoint2` geben die zwei nächstgelegenen Punkte aus den beiden Punktwolken zurück.

Anwendung

Das Bestimmen der minimalen Distanz zweier Linien sieht z. B. so aus:

```
hr := F_VN_ClosestPointsBF(
    ipPointSet1 :=  ipLine1,
    ipPointSet2 :=  ipLine2,
    fMinDist    :=  fDistance,
    aPoint1     :=  aPoint1,
```

```
    aPoint2     :=  aPoint2,
    hrPrev      :=  hr
);
```

Verwandte Funktionen

- ClosestPointsBF [▶ 1518] zur Berechnung der nächstgelegenen Punkte zweier Punktewolken
- MeasureAngleBetweenEdges [▶ 1544] zur Vermessung des Winkels zwischen zwei Kanten
- MeasureEdgeDistance [▶ 1549] zur Vermessung des durchschnittlichen Abstands zweier Kanten
- MeasureMinEdgeDistance [▶ 1554] zur Vermessung des minimalen Abstands zweier Kanten

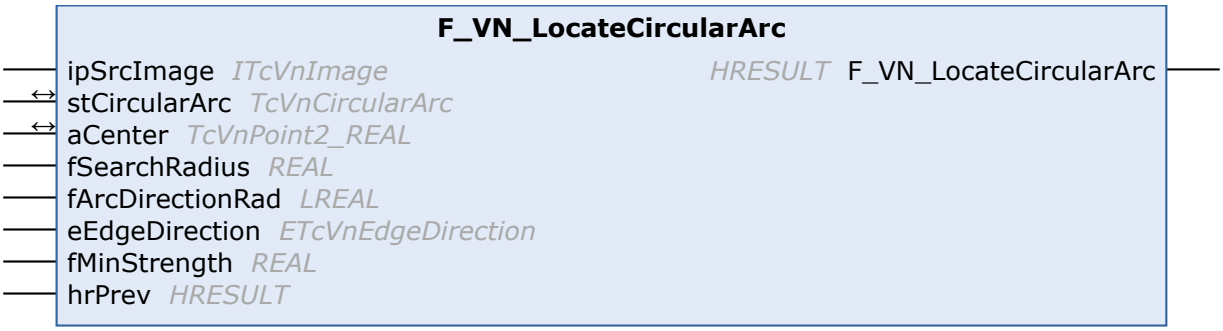
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.20.3 F_VN_LocateCircularArc



Locate a circular arc, using an interpolation method.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_LocateCircularArc : HRESULT
VAR_INPUT
    ipSrcImage       : ITcVnImage;
END_VAR
VAR_IN_OUT
    stCircularArc    : TcVnCircularArc;
    aCenter          : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    fSearchRadius    : REAL;
    fArcDirectionRad : LREAL;
    eEdgeDirection   : ETcVnEdgeDirection;
    fMinStrength     : REAL;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel)
fSearchRadius	REAL	Input search radius (starting from aCenter, should be greater than the actual circle radius but aCenter + fSearchRadius should be within the image borders)
fArcDirectionRad	LREAL	Input search starting direction in radian. Valid range is [-pi, +pi], where right is 0 rad, top is -pi/2 rad and bottom +pi/2 rad. The circular arc should at least be valid in range fArcDirectionRad +- 0.4 rad.
eEdgeDirection	ETcVnEdgeDirection [▶ 210]	Specification of the edge direction to search for
fMinStrength	REAL	Specification of the minimum strength (intensity difference) of the edge to search for
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stCircularArc	TcVnCircularArc [▶ 241]	Returns the circular arc parameters
aCenter	TcVnPoint2_REAL [▶ 171]	Input estimated circle center (only used as a starting point to search for the circle contour, not used for the circle center estimation)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_LocateCircularArc` lokalisiert und misst einen anteiligen Kreisbogen. Sie basiert auf der Kantenlokalisierung [▶ 1512] und nutzt übliche Parameter. Sie gibt den lokalisierten Kreisbogen als Struktur vom Typ TcVnCircularArc (Mittelpunkt, Radius, Winkelbereich) zurück.

Falls anstatt eines Kreisbogens vollständige Kreise vermessen werden sollen, verwenden Sie stattdessen die Funktion F_VN_LocateEllipse [▶ 1538].

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Grauwertbild sein.

Kreisbogen (Rückgabewert)

Der Parameter `stCircularArc` gibt den lokalisierten Kreisbogen als Struktur vom Typ TcVnCircularArc [▶ 241] zurück.

Geschätzter Mittelpunkt

Der geschätzte Mittelpunkt `aCenter` des Kreisbogens muss angegeben werden, um die Position der Suchlinien zu definieren.

Falls die Position des Kreismittelpunkts unbekannt ist, muss sie ggfs. vorher mittels einer Objektdetektion [▶ 1205] ermittelt werden.

Suchradius

Der Suchradius `fSearchRadius` legt die Länge der Suchlinien vom Kreismittelpunkt aus in Pixel fest. Der Suchradius sollte etwas größer als der tatsächlich Kreisradius definiert werden, damit die Kantenlokalisierung eindeutig ausgeführt werden kann.

Suchrichtung

Die Suchrichtung `fArcDirectionRad` legt fest, in welcher Kreisrichtung mit der Kantensuche begonnen werden soll (ausgehend vom Kreismittelpunkt). Der Wert ist in Radiant anzugeben und hat einen Wertebereich von [-π, π]. Dabei korrespondiert rechts mit 0, oben mit –π/2 und unten mit π/2. Der Kreisbogen muss von `fArcDirectionRad` aus mindestens in Bereich von +- 0,4 vorhanden sein, um erkannt zu werden. Es wird empfohlen, die Mitte des Kreisbogens anzuvisieren.

Parameter zur Kantenlokalisierung

Die restlichen Parameter sind ausführlich im Kapitel Kantenlokalisierung [▶ 1512] erklärt.

Expert-Parameter

Die Expert-Variante F_VN_LocateCircularArcExp [▶ 1523] enthält zusätzliche Parameter.

Anwendung

Die Lokalisierung und Vermessung eines anteiligen Kreisbogens mit einem Radius von `270px` und einem Suchwinkel von `120°` sieht z. B. so aus:

```
VAR
    stArc   :   TcVnCircularArc;
    aCenter :   TcVnPoint2_REAL := [420, 310];
END_VAR

hr := F_VN_LocateCircularArc(
    ipSrcImage          :=  ipImageIn,
    stCircularArc       :=  stArc,
    aCenterPoint        :=  aCenter,
    fSearchRadius       :=  270,
    fArcDirectionRad    :=  2.1, // 2.1 rad equals 120°
    eEdgeDirection      :=  TCVN_ED_LIGHT_TO_DARK,
    fMinStrength        :=  100,
    hrPrev              :=  hr
);
```

Die Eigenschaften des lokalisierten Kreisbogens können dann über die Variablen `stArc.aCenter`, `stArc.fRadius`, `stArc.fStartAngle` und `stArc.fEndAngle` abgerufen werden. Zudem kann der Kreisbogen mit der Funktion F_VN_DrawCircularArc [▶ 1035] visualisiert werden:

```
F_VN_DrawCircularArc(stArc, ipImageRes, aColorGreen, 5, hr);
```

Beispiele

- Locate Circular Arc [▶ 2900]

Verwandte Funktionen

- LocateCircularArc [▶ 1520] zur Lokalisierung von anteiligen Kreisbögen
- LocateEdge [▶ 1526] zur Lokalisierung von linearen Kanten
- LocateEdges [▶ 1532] zur Lokalisierung von mehreren linearen Kanten
- LocateEllipse [▶ 1538] zur Lokalisierung von vollständigen Ellipsen

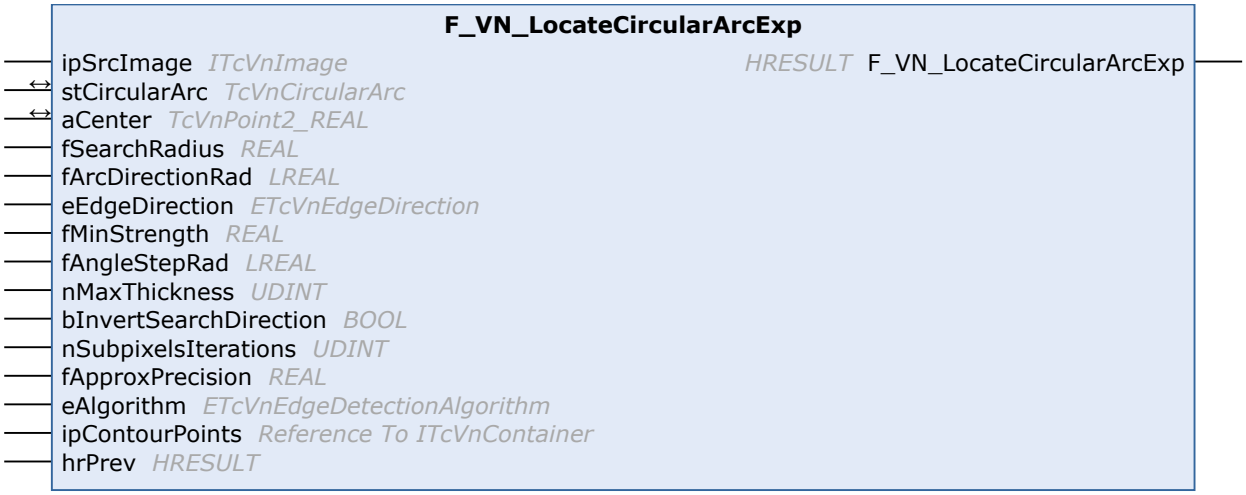
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.20.4 F_VN_LocateCircularArcExp



Locate a circular arc. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_LocateCircularArcExp : HRESULT
VAR_INPUT
    ipSrcImage             : ITcVnImage;
END_VAR
VAR_IN_OUT
    stCircularArc          : TcVnCircularArc;
    aCenter                : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    fSearchRadius          : REAL;
    fArcDirectionRad       : LREAL;
    eEdgeDirection         : ETcVnEdgeDirection;
    fMinStrength           : REAL;
    fAngleStepRad          : LREAL;
    nMaxThickness          : UDINT;
    bInvertSearchDirection : BOOL;
    nSubpixelsIterations   : UDINT;
    fApproxPrecision       : REAL;
    eAlgorithm             : ETcVnEdgeDetectionAlgorithm;
    ipContourPoints        : Reference To ITcVnContainer;
    hrPrev                 : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel)
fSearchRadius	REAL	Input search radius (starting from aCenter, should be greater than the actual circle radius but aCenter + fSearchRadius should be within the image borders)
fArcDirectionRad	LREAL	Input search starting direction in radian. Valid range is [-pi, +pi], where right is 0 rad, top is -pi/2 rad and bottom +pi/2 rad. The circular arc should at least be valid in range fArcDirectionRad +- 4 * fAngleStepRad)
eEdgeDirection	ETcVnEdgeDirection [▶ 210]	Specification of the edge direction to search for
fMinStrength	REAL	Specification of the minimum strength (intensity difference) of the edge to search for
fAngleStepRad	LREAL	Search step in rad (should be chosen so that about 20 - 60 steps are available for the whole arc. In most cases 0.1 rad (5.7 deg) is a good value)
nMaxThickness	UDINT	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels
bInvertSearchDirection	BOOL	If true, the search starts from outside the circular arc in direction of the center
nSubpixelsIterations	UDINT	Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)
fApproxPrecision	REAL	Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)
eAlgorithm	ETcVnEdgeDetectionAlgorithm [▶ 209]	Selection of the edge detection algorithm
ipContourPoints	Reference To ITcVnContainer [▶ 385]	Returns the subpixel accurate contour (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_REAL; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stCircularArc	TcVnCircularArc [▶ 241]	Returns the circular arc parameters
aCenter	TcVnPoint2_REAL [▶ 171]	Input estimated circle center (only used as a starting point to search for the circle contour, not used for the circle center estimation)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_LocateCircularArcExp` ist die Expert-Variante von F_VN_LocateCircularArc [▶ 1520]. Sie enthält zusätzliche Parameter.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Grauwertbild sein.

Kreisbogen (Rückgabewert)

Der Parameter `stCircularArc` gibt den lokalisierten Kreisbogen als Struktur vom Typ TcVnCircularArc [▶ 241] zurück.

Geschätzter Mittelpunkt

Der geschätzte Mittelpunkt `aCenter` des Kreisbogens muss angegeben werden, um die Position der Suchlinien zu definieren.

Falls die Position des Kreismittelpunkts unbekannt ist, muss sie ggfs. vorher mittels einer Objektdetektion [▶ 1205] ermittelt werden.

Suchradius

Der Suchradius `fSearchRadius` legt die Länge der Suchlinien vom Kreismittelpunkt aus in Pixel fest. Der Suchradius sollte etwas größer als der tatsächlich Kreisradius definiert werden, damit die Kantenlokalisierung eindeutig ausgeführt werden kann.

Suchrichtung

Die Suchrichtung `fArcDirectionRad` legt fest, in welcher Kreisrichtung mit der Kantensuche begonnen werden soll (ausgehend vom Kreismittelpunkt). Der Wert ist in Radiant anzugeben und hat einen Wertebereich von [-π, π]. Dabei korrespondiert rechts mit 0, oben mit –π/2 und unten mit π/2. Der Kreisbogen muss von `fArcDirectionRad` aus mindestens in Bereich von +- 0,4 vorhanden sein, um erkannt zu werden. Es wird empfohlen, die Mitte des Kreisbogens anzuvisieren.

Radialer Suchlinien-Abstand (Expert)

Der radiale Suchlinien-Abstand `fAngleStepRad` legt fest, wie groß die Winkelschritte zwischen den Suchlinien sind. Er wird in Radiant angegeben. Der Abstand sollte so gewählt werden, dass der gesamte Kreisbogen in etwa 20 - 60 Schritte aufgeteilt ist. In den meisten Fällen ist `0,1 rad (5,7°)` ein passender Wert.

Suchrichtungs-Invertierung (Expert)

Mit dem Boolean `bInvertSearchDirection` kann die Suchrichtung invertiert werden. In diesem Fall heißt dies folgendes:

- `FALSE`: Suche vom Kreismittelpunkt nach außerhalb des Kreisbogens.
- `TRUE`: Suche von außerhalb des Kreisbogens zum Kreismittelpunkt

Kontur-Punkte (Expert, Rückgabewert)

Der Container `ipContourPoints` gibt die gefundenen Kantenpunkte des Kreisbogens zurück und ist daher vom Typ `ContainerType_Vector_TcVnPoint2_REAL`. Dabei enthält der Container je einen Punkt pro erfolgreich gefundener Suchlinie.

Wenn die Rückgabe der Kantenpunkte nicht benötigt wird, kann der Parameter auf `0` gesetzt werden.

Kreisbogen stCircularArc

Aus der Menge dieser Punkte wird der Kreisbogen stCircularArc approximiert.

Parameter zur Kantenlokalisierung

Die restlichen Parameter sind ausführlich im Kapitel Kantenlokalisierung [▶ 1512] erklärt.

Anwendung

Die Lokalisierung eines Kreisbogens mit Expert-Parametern sieht z.B. so aus:

```
hr := F_VN_LocateCircularArc(
    ipSrcImage          :=  ipImageIn,
    stCircularArc       :=  stArc,
    aCenterPoint        :=  aCenter,
    fSearchRadius       :=  270,
    fArcDirectionRad    :=  2.1,
    eEdgeDirection      :=  TCVN_ED_LIGHT_TO_DARK,
    fMinStrength        :=  100,
    fAngleStepRad       :=  40,
    nMaxThickness       :=  10,
    bInvertSearchDirection :=  FALSE,
    nSubpixelsIterations   :=  100,
    fApproxPrecision    :=  0.0001,
    eAlgorithm          :=  TCVN_EDA_APPROX_ERF,
    ipContourPoints     :=  ipEllipseContour,
    hrPrev              :=  hr
);
```

Die lokalisierten Punkte können mit der Funktion F_VN_DrawContours [▶ 1040] visualisiert werden.

```
hr := F_VN_DrawContours(ipEllipseContour, -1, ipImageRes, aColorGreen, 5, hr);
```

Beispiele

- Locate Circular Arc [▶ 2900]

Verwandte Funktionen

- LocateCircularArc [▶ 1520] zur Lokalisierung von anteiligen Kreisbögen
- LocateEdge [▶ 1526] zur Lokalisierung von linearen Kanten
- LocateEdges [▶ 1532] zur Lokalisierung von mehreren linearen Kanten
- LocateEllipse [▶ 1538] zur Lokalisierung von vollständigen Ellipsen

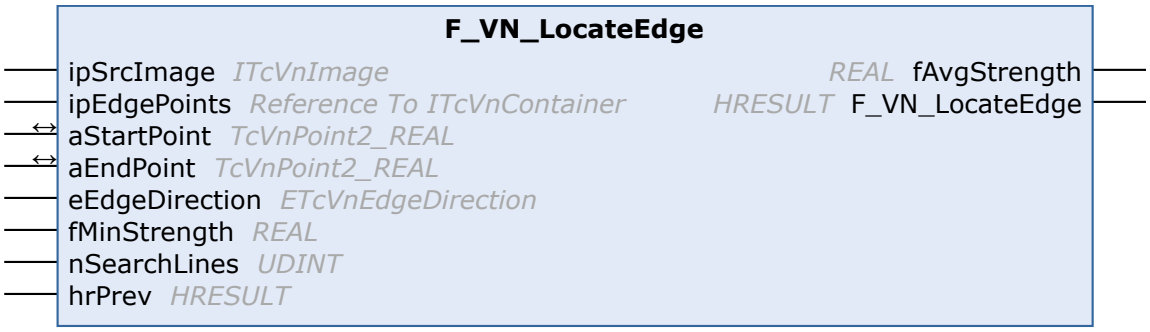
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.20.5 F_VN_LocateEdge



Locate the points of the first occuring edge inside a specified search window, using an interpolation method.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_LocateEdge : HRESULT
VAR_INPUT
    ipSrcImage     : ITcVnImage;
    ipEdgePoints   : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    aStartPoint    : TcVnPoint2_REAL;
    aEndPoint      : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    eEdgeDirection : ETcVnEdgeDirection;
    fMinStrength   : REAL;
    nSearchLines   : UDINT;
    hrPrev         : HRESULT;
END_VAR
VAR_OUTPUT
    fAvgStrength   : REAL;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel)
ipEdgePoints	Reference To ITcVnContainer [▶ 385]	Returns the detected edge points (ContainerType_Vector_TcVnPoint2_REAL)
eEdgeDirection	ETcVnEdgeDirection [▶ 210]	Specification of the edge direction to search for
fMinStrength	REAL	Specification of the minimum strength (intensity difference) of the edge to search for
nSearchLines	UDINT	Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ...), centered around the line specified by aStartPoint and aEndPoint
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aStartPoint	TcVnPoint2_REAL [▶ 171]	Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 171]	Position where the search process ends

Outputs

Name	Type	Description
fAvgStrength	REAL	Returns the average strength of the detected edge

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_LocateEdge` lokalisiert eine Kante in einem definierten Suchfenster.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Grauwertbild sein.

Lokalisierte Punkte (Rückgabewert)

Der Container `ipEdgePoints` gibt die lokalisierten Punkte der linearen Kante zurück.

Das Suchfenster

Zunächst einmal muss ein Suchfenster definiert werden, um festzulegen wo und in welcher Richtung nach einer Kante gesucht werden soll. Dies geschieht mit den Parametern aStartPoint, aEndPoint und nSearchLines. Das Suchfenster ergibt sich aus der Suchlinie von aStartPoint nach aEndPoint und parallel dazu (nSearchLines - 1) / 2 Suchlinien in jede Richtung, mit jeweils 1 Pixel Abstand. Somit definiert nSearchLines die Anzahl der Suchlinien und muss immer einen ungeraden Wert haben.

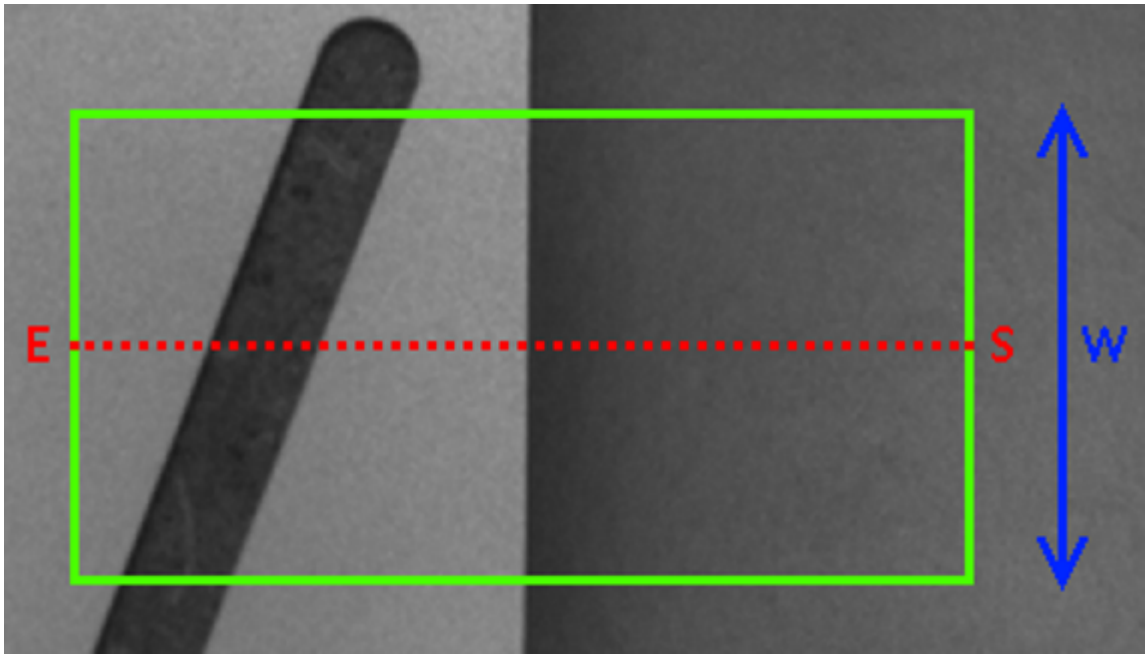


Abb. 15: S: StartPoint, E: EndPoint, W: WindowWidth

Kantenstärke (Rückgabewert)

Der Rückgabewert `fAvgStrength` zeigt den tatsächlichen Kontrast der lokalisierten Kante (gemittelt über alle Suchlinien) an.

Parameter zur Kantenlokalisierung

Die restlichen Parameter sind ausführlich im Kapitel Kantenlokalisierung [▶ 1512] erklärt.

Expert-Parameter

Die Expert-Variante F_VN_LocateEdgeExp enthält zusätzliche Parameter.

Anwendung

Die Lokalisierung einer linearen Kante sieht z. B. so aus:

```
hr := F_VN_LocateEdge(
    ipSrcImage              :=  ipImageIn,
    ipEdgePoints            :=  ipEdgePoints,
    aStartPoint             :=  aStartPoint,
    aEndPoint               :=  aEndPoint,
    eEdgeDirection          :=  eDirection,
    fMinStrength            :=  fMinStrength,
    nSearchLines            :=  nSearchLines,
```

```
    hrPrev                  :=  hr,
    fAvgStrength            =>  fAvgStrength
);
```

Die lokalisierten Kantenpunkte können mit der Funktion F_VN_DrawContours [▶ 1040] visualisiert werden:

```
F_VN_DrawContours(ipEdgePoints, -1, ipImageRes, aColorGreen, 5, hr);
```

Beispiele

- Locate Edge [▶ 2904]

Verwandte Funktionen

- LocateCircularArc [▶ 1520] zur Lokalisierung von anteiligen Kreisbögen
- LocateEdge [▶ 1526] zur Lokalisierung von linearen Kanten
- LocateEdges [▶ 1532] zur Lokalisierung von mehreren linearen Kanten
- LocateEllipse [▶ 1538] zur Lokalisierung von vollständigen Ellipsen

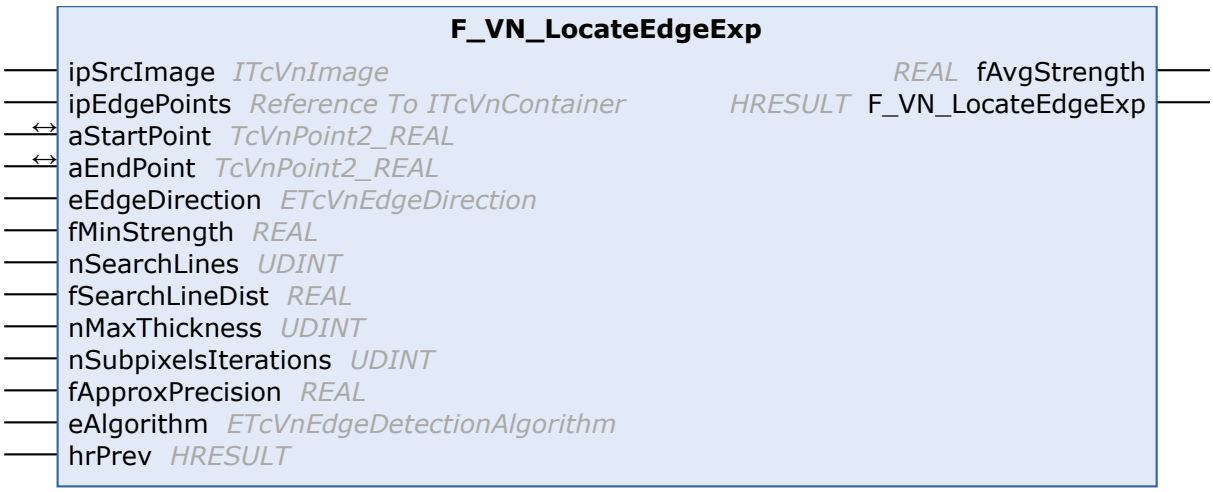
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.20.6 F_VN_LocateEdgeExp



Locate the points of the first occuring edge inside a specified search window with subpixel accuracy. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_LocateEdgeExp : HRESULT
VAR_INPUT
    ipSrcImage          : ITcVnImage;
    ipEdgePoints        : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    aStartPoint         : TcVnPoint2_REAL;
    aEndPoint           : TcVnPoint2_REAL;
```

```
END_VAR
VAR_INPUT
    eEdgeDirection       : ETcVnEdgeDirection;
    fMinStrength         : REAL;
    nSearchLines         : UDINT;
    fSearchLineDist      : REAL;
    nMaxThickness        : UDINT;
    nSubpixelsIterations : UDINT;
    fApproxPrecision     : REAL;
    eAlgorithm           : ETcVnEdgeDetectionAlgorithm;
    hrPrev               : HRESULT;
END_VAR
VAR_OUTPUT
    fAvgStrength         : REAL;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel)
ipEdgePoints	Reference To ITcVnContainer [▶ 385]	Returns the detected edge points (ContainerType_Vector_TcVnPoint2_REAL)
eEdgeDirection	ETcVnEdgeDirection [▶ 210]	Specification of the edge direction to search for
fMinStrength	REAL	Specification of the minimum strength (intensity difference) of the edge to search for
nSearchLines	UDINT	Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ...), centered around the line specified by aStartPoint and aEndPoint
fSearchLineDist	REAL	Distance between the search lines in pixels (> 0)
nMaxThickness	UDINT	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels
nSubpixelsIterations	UDINT	Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)
fApproxPrecision	REAL	Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)
eAlgorithm	ETcVnEdgeDetectionAlgorithm [▶ 209]	Selection of the edge detection algorithm
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aStartPoint	TcVnPoint2_REAL [▶ 171]	Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 171]	Position where the search process ends

Outputs

Name	Type	Description
fAvgStrength	REAL	Returns the average strength of the detected edge

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_LocateEdgeExp` ist die Expert-Variante von F_VN_LocateEdge [▶ 1526]. Sie enthält zusätzliche Parameter.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Grauwertbild sein.

Lokalisierte Punkte (Rückgabewert)

Der Container `ipEdgePoints` gibt die lokalisierten Punkte der linearen Kante zurück.

Das Suchfenster

Zunächst einmal muss ein Suchfenster definiert werden, um festzulegen wo und in welcher Richtung nach einer Kante gesucht werden soll. Dies geschieht mit den Parametern aStartPoint, aEndPoint und nSearchLines. Das Suchfenster ergibt sich aus der Suchlinie von aStartPoint nach aEndPoint und parallel dazu (nSearchLines - 1) / 2 Suchlinien in jede Richtung, mit jeweils 1 Pixel Abstand. Somit definiert nSearchLines die Anzahl der Suchlinien und muss immer einen ungeraden Wert haben.

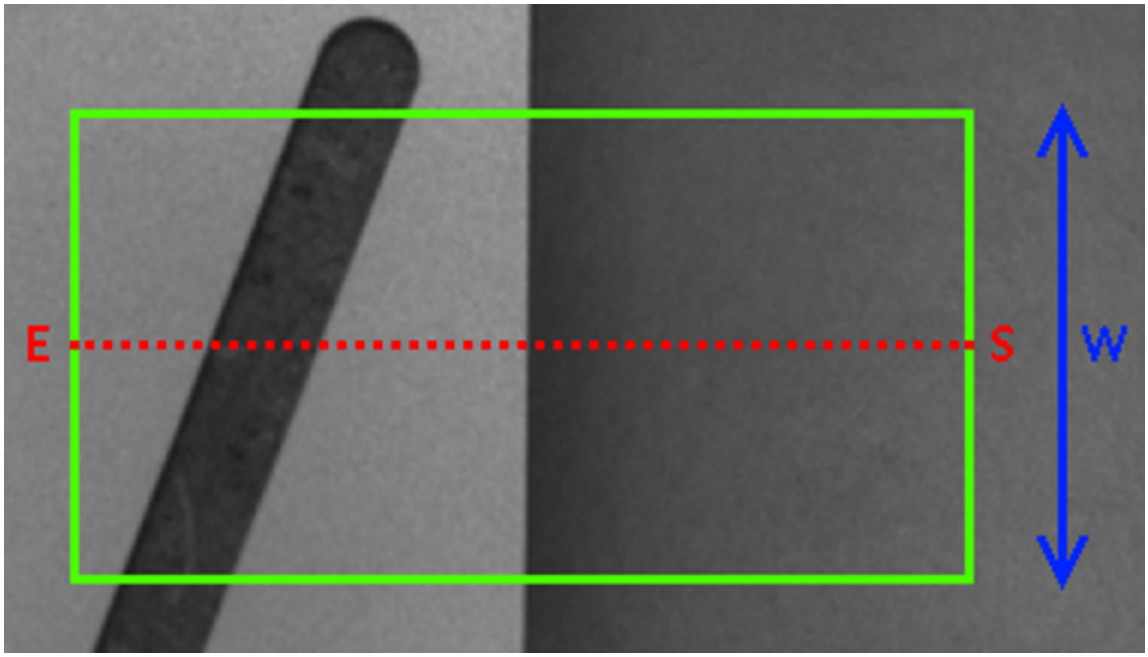


Abb. 16: S: StartPoint, E: EndPoint, W: WindowWidth

Kantenstärke (Rückgabewert)

Der Rückgabewert `fAvgStrength` zeigt den tatsächlichen Kontrast der lokalisierten Kante (gemittelt über alle Suchlinien) an.

Parameter zur Kantenlokalisierung

Die restlichen Parameter sind ausführlich im Kapitel Kantenlokalisierung [▶ 1512] erklärt.

Anwendung

Die Lokalisierung einer linearen Kante mit Expert-Parametern sieht z. B. so aus:

```
hr := F_VN_LocateEdgeExp(
    ipSrcImage              :=  ipImageIn,
    ipEdgePoints            :=  ipEdgePoints,
    aStartPoint             :=  aStartPoint,
    aEndPoint               :=  aEndPoint,
    eEdgeDirection          :=  eDirection,
    fMinStrength            :=  fMinStrength,
    nSearchLines            :=  nSearchLines,
    fSearchLineDist         :=  fSearchLineDist
    nMaxThickness           :=  nMaxThickness,
    nSubpixelsIterations    :=  nSubpixIter,
    fApproxPrecision        :=  0.0001,
    eAlgorithm              :=  eAlgorithm,
    hrPrev                  :=  hr,
    fAvgStrength            =>  fAvgStrength
);
```

Beispiele

- Locate Edge [▶ 2904]

Verwandte Funktionen

- LocateCircularArc [▶ 1520] zur Lokalisierung von anteiligen Kreisbögen
- LocateEdge [▶ 1526] zur Lokalisierung von linearen Kanten
- LocateEdges [▶ 1532] zur Lokalisierung von mehreren linearen Kanten
- LocateEllipse [▶ 1538] zur Lokalisierung von vollständigen Ellipsen

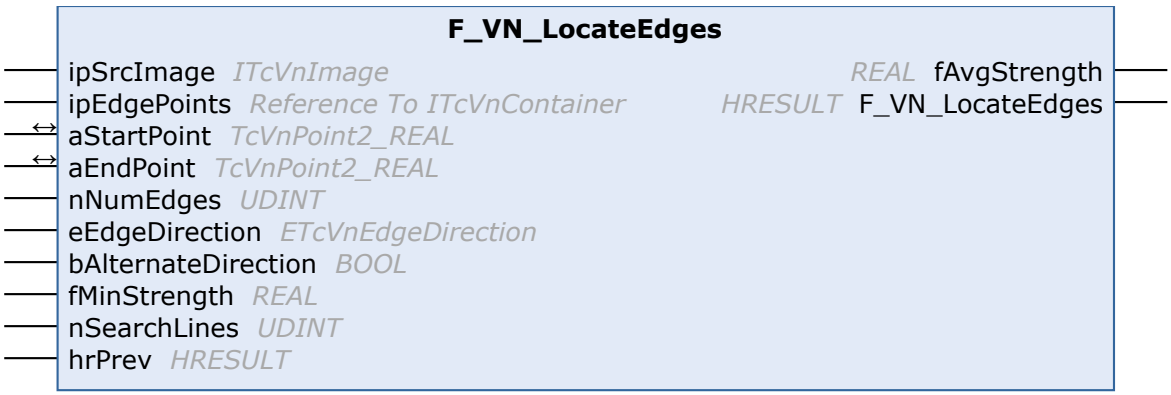
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.20.7 F_VN_LocateEdges



Locate the points of multiple occuring edges inside a specified search window, using an interpolation method.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_LocateEdges : HRESULT
VAR_INPUT
    ipSrcImage          : ITcVnImage;
```

```
    ipEdgePoints        : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    aStartPoint         : TcVnPoint2_REAL;
    aEndPoint           : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    nNumEdges           : UDINT;
    eEdgeDirection      : ETcVnEdgeDirection;
    bAlternateDirection : BOOL;
    fMinStrength        : REAL;
    nSearchLines        : UDINT;
    hrPrev              : HRESULT;
END_VAR
VAR_OUTPUT
    fAvgStrength        : REAL;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel)
ipEdgePoints	Reference To ITcVnContainer [▶ 385]	Returns the detected edge points (ContainerType_Vector_Vector_TcVnPoint2_REAL)
nNumEdges	UDINT	The (maximum) number of edges to search for
eEdgeDirection	ETcVnEdgeDirection [▶ 210]	Specification of the edge direction to search for
bAlternateDirection	BOOL	If true, eEdgeDirection is alternated after each detected edge. Else, only edges with eEdgeDirection are searched for.
fMinStrength	REAL	Specification of the minimum strength (intensity difference) of the edge to search for
nSearchLines	UDINT	Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ...), centered around the line specified by aStartPoint and aEndPoint
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aStartPoint	TcVnPoint2_REAL [▶ 171]	Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 171]	Position where the search process ends

Outputs

Name	Type	Description
fAvgStrength	REAL	Returns the average strength of the detected edges

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion F_VN_LocateEdges lokalisiert mehrere Kanten in einem definierten Suchfenster.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Grauwertbild sein.

Lokalisierte Punkte (Rückgabewert)

Der Container `ipEdgePoints` gibt die lokalisierten Punkte der linearen Kanten zurück.

Das Suchfenster

Zunächst einmal muss ein Suchfenster definiert werden, um festzulegen wo und in welcher Richtung nach einer Kante gesucht werden soll. Dies geschieht mit den Parametern aStartPoint, aEndPoint und nSearchLines. Das Suchfenster ergibt sich aus der Suchlinie von aStartPoint nach aEndPoint und parallel dazu (nSearchLines - 1) / 2 Suchlinien in jede Richtung, mit jeweils 1 Pixel Abstand. Somit definiert nSearchLines die Anzahl der Suchlinien und muss immer einen ungeraden Wert haben.

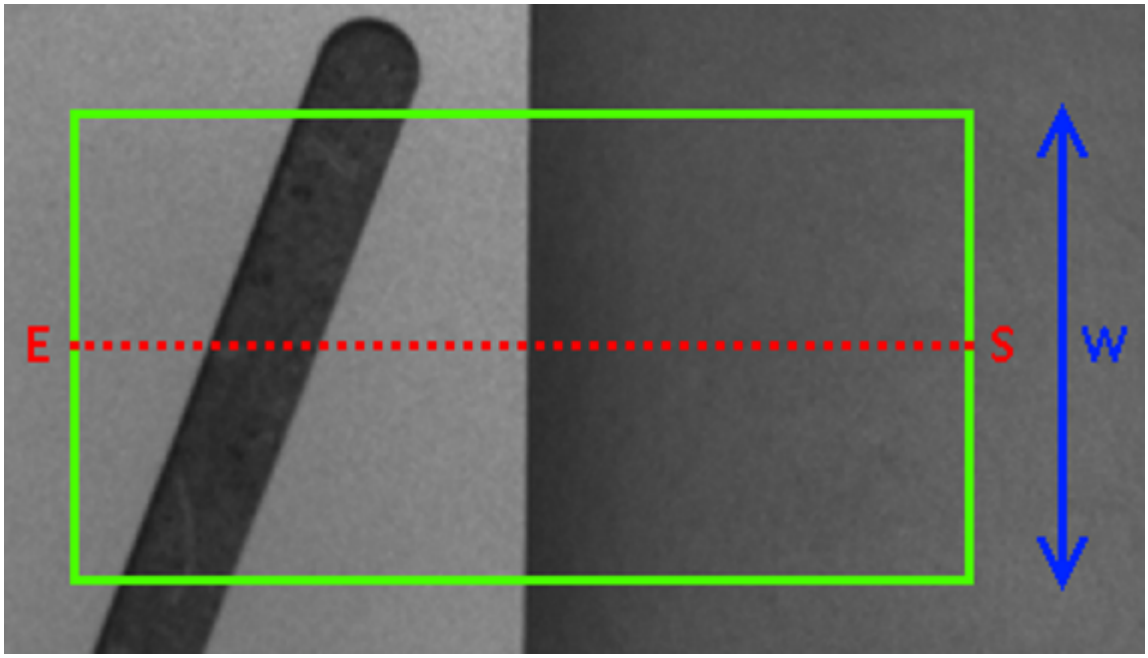


Abb. 17: S: StartPoint, E: EndPoint, W: WindowWidth

Kantenstärke (Rückgabewert)

Der Rückgabewert `fAvgStrength` zeigt den tatsächlichen Kontrast der lokalisierten Kanten (gemittelt über alle Suchlinien) an.

Parameter zur Kantenlokalisierung

Die restlichen Parameter sind ausführlich im Kapitel Kantenlokalisierung [▶ 1512] erklärt.

Expert-Parameter

Die Expert-Variante F_VN_LocateEdgeExp enthält zusätzliche Parameter.

Anwendung

Die Lokalisierung einer linearen Kante sieht z. B. so aus:

```
hr := F_VN_LocateEdges(
    ipSrcImage              :=  ipImageIn,
    ipEdgePoints            :=  ipEdgePoints,
    aStartPoint             :=  aStartPoint,
    aEndPoint               :=  aEndPoint,
    nNumEdges               :=  nNumEdges,
    eEdgeDirection          :=  eDirection,
    bAlternateDirection     :=  bAlternateDirection,
    fMinStrength            :=  fMinStrength,
    nSearchLines            :=  nSearchLines,
```

```
    hrPrev                  :=  hr,
    fAvgStrength            =>  fAvgStrength
);
```

Die lokalisierten Kantenpunkte können mit der Funktion F_VN_DrawContours [▶ 1040] visualisiert werden:

```
F_VN_DrawContours(ipEdgePoints, -1, ipImageRes, aColorGreen, 5, hr);
```

Verwandte Funktionen

- LocateCircularArc [▶ 1520] zur Lokalisierung von anteiligen Kreisbögen
- LocateEdge [▶ 1526] zur Lokalisierung von linearen Kanten
- LocateEdges [▶ 1532] zur Lokalisierung von mehreren linearen Kanten
- LocateEllipse [▶ 1538] zur Lokalisierung von vollständigen Ellipsen

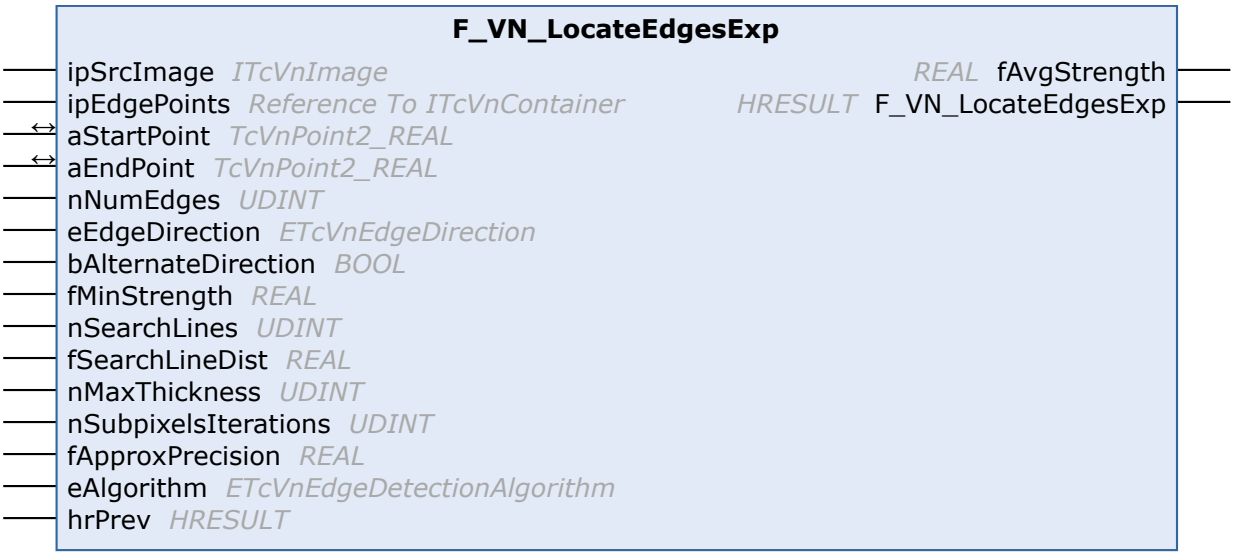
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.20.8 F_VN_LocateEdgesExp



Locate the points of multiple occuring edges inside a specified search window with subpixel accuracy. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_LocateEdgesExp : HRESULT
VAR_INPUT
    ipSrcImage          : ITcVnImage;
    ipEdgePoints        : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    aStartPoint         : TcVnPoint2_REAL;
    aEndPoint           : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
```

```
    nNumEdges            : UDINT;
    eEdgeDirection       : ETcVnEdgeDirection;
    bAlternateDirection  : BOOL;
    fMinStrength         : REAL;
    nSearchLines         : UDINT;
    fSearchLineDist      : REAL;
    nMaxThickness        : UDINT;
    nSubpixelsIterations : UDINT;
    fApproxPrecision     : REAL;
    eAlgorithm           : ETcVnEdgeDetectionAlgorithm;
    hrPrev               : HRESULT;
END_VAR
VAR_OUTPUT
    fAvgStrength         : REAL;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel)
ipEdgePoints	Reference To ITcVnContainer [▶ 385]	Returns the detected edge points (ContainerType_Vector_Vector_TcVnPoint2_REAL)
nNumEdges	UDINT	The (maximum) number of edges to search for
eEdgeDirection	ETcVnEdgeDirection [▶ 210]	Specification of the edge direction to search for
bAlternateDirection	BOOL	If true, eEdgeDirection is alternated after each detected edge. Else, only edges with eEdgeDirection are searched for.
fMinStrength	REAL	Specification of the minimum strength (intensity difference) of the edge to search for
nSearchLines	UDINT	Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ...), centered around the line specified by aStartPoint and aEndPoint
fSearchLineDist	REAL	Distance between the search lines in pixels (> 0)
nMaxThickness	UDINT	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels
nSubpixelsIterations	UDINT	Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)
fApproxPrecision	REAL	Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)
eAlgorithm	ETcVnEdgeDetectionAlgorithm [▶ 209]	Selection of the edge detection algorithm
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aStartPoint	TcVnPoint2_REAL [▶ 171]	Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 171]	Position where the search process ends

Outputs

Name	Type	Description
fAvgStrength	REAL	Returns the average strength of the detected edges

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion F_VN_LocateEdgesExp ist die Expert-Variante von F_VN_LocateEdges [▶ 1532]. Sie enthält zusätzliche Parameter.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Grauwertbild sein.

Lokalisierte Punkte (Rückgabewert)

Der Container `ipEdgePoints` gibt die lokalisierten Punkte der linearen Kanten zurück.

Das Suchfenster

Zunächst einmal muss ein Suchfenster definiert werden, um festzulegen wo und in welcher Richtung nach einer Kante gesucht werden soll. Dies geschieht mit den Parametern aStartPoint, aEndPoint und nSearchLines. Das Suchfenster ergibt sich aus der Suchlinie von aStartPoint nach aEndPoint und parallel dazu (nSearchLines - 1) / 2 Suchlinien in jede Richtung, mit jeweils 1 Pixel Abstand. Somit definiert nSearchLines die Anzahl der Suchlinien und muss immer einen ungeraden Wert haben.

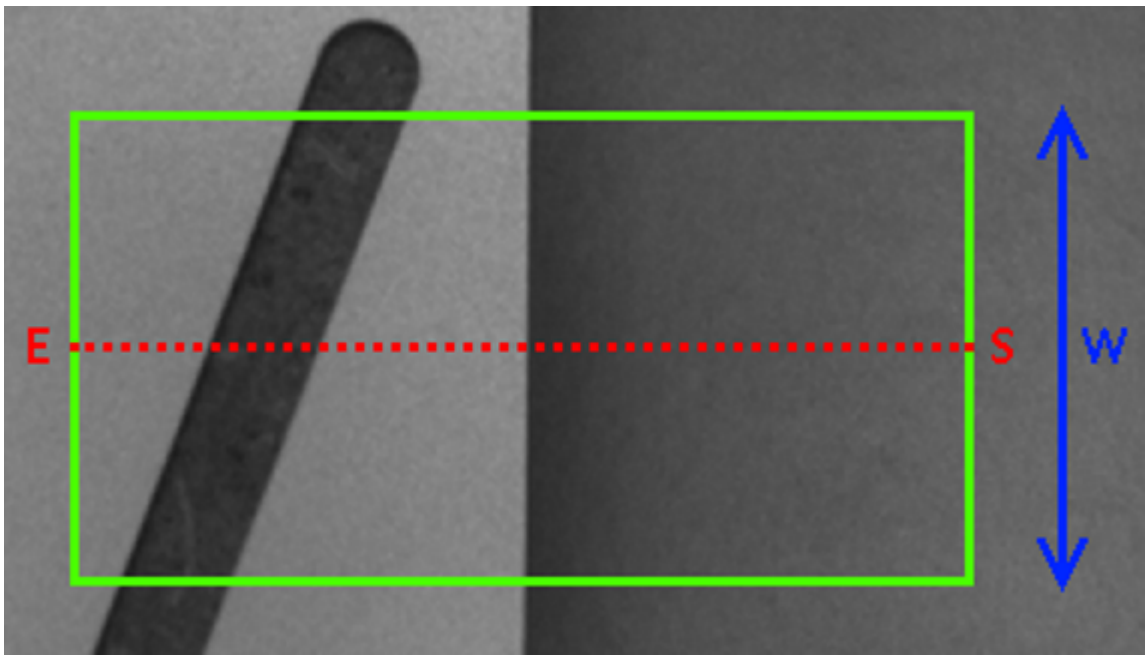


Abb. 18: S: StartPoint, E: EndPoint, W: WindowWidth

Kantenstärke (Rückgabewert)

Der Rückgabewert `fAvgStrength` zeigt den tatsächlichen Kontrast der lokalisierten Kanten (gemittelt über alle Suchlinien) an.

Parameter zur Kantenlokalisierung

Die restlichen Parameter sind ausführlich im Kapitel Kantenlokalisierung [▶ 1512] erklärt.

Anwendung

Die Lokalisierung einer linearen Kante mit Expert-Parametern sieht z. B. so aus:

```
hr := F_VN_LocateEdgesExp(
    ipSrcImage              :=  ipImageIn,
    ipEdgePoints            :=  ipEdgePoints,
    aStartPoint             :=  aStartPoint,
    aEndPoint               :=  aEndPoint,
    nNumEdges               :=  nNumEdges,
    eEdgeDirection          :=  eDirection,
    bAlternateDirection     :=  bAlternateDirection,
    fMinStrength            :=  fMinStrength,
    nSearchLines            :=  nSearchLines,
    fSearchLineDist         :=  fSearchLineDist
    nMaxThickness           :=  nMaxThickness,
    nSubpixelsIterations    :=  nSubpixIter,
    fApproxPrecision        :=  0.001,
    eAlgorithm              :=  eAlgorithm,
    hrPrev                  :=  hr,
    fAvgStrength            =>  fAvgStrength
);
```

Verwandte Funktionen

- LocateCircularArc [▶ 1520] zur Lokalisierung von anteiligen Kreisbögen
- LocateEdge [▶ 1526] zur Lokalisierung von linearen Kanten
- LocateEdges [▶ 1532] zur Lokalisierung von mehreren linearen Kanten
- LocateEllipse [▶ 1538] zur Lokalisierung von vollständigen Ellipsen

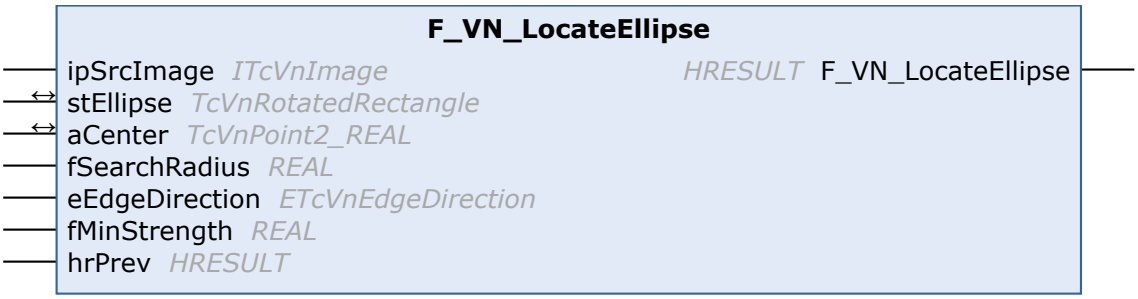
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.20.9 F_VN_LocateEllipse



Locate an ellipse, using an interpolation method for locating the edges.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_LocateEllipse : HRESULT
VAR_INPUT
    ipSrcImage      : ITcVnImage;
END_VAR
```

```
VAR_IN_OUT
    stEllipse      : TcVnRotatedRectangle;
    aCenter        : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    fSearchRadius  : REAL;
    eEdgeDirection : ETcVnEdgeDirection;
    fMinStrength   : REAL;
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel)
fSearchRadius	REAL	Radius around aCenter to search for edges (aCenter + fSearchRadius in all directions must be inside image borders!)
eEdgeDirection	ETcVnEdgeDirection [▶ 210]	Specification of the edge direction to search for (from center to outside ellipse)
fMinStrength	REAL	Specification of the minimum strength (intensity difference) of the edge to search for
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stEllipse	TcVnRotatedRectangle [▶ 258]	Returns the detected ellipse
aCenter	TcVnPoint2_REAL [▶ 171]	The expected ellipse center

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_LocateEllipse` lokalisiert und misst eine Ellipse. Sie basiert auf der Kantenlokalisierung [▶ 1512] und nutzt übliche Parameter. Sie gibt die lokalisierte Ellipse als Struktur vom Typ TcVnRotatedRectangle [▶ 258] (Mittelpunkt, Größe und Winkel) zurück.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Grauwertbild sein.

Ellipse (Rückgabewert)

Der Parameter stEllipse liefert die lokalisierte Ellipse als Struktur vom Typ TcVnRotatedRectangle [▶ 258] zurück (Mittelpunkt, Größe und Winkel).

Suchkreis

Der kreisförmige Suchbereich wird über den erwarteten Mittelpunkt `aCenter` und sowie einen Suchradius `fSearchRadius` definiert. Der Suchradius sollte etwas größer als der erwartete Ellipsenradius festgelegt werden, um die Ellipse sicher zu lokalisieren.

Parameter zur Kantenlokalisierung

Die restlichen Parameter sind ausführlich im Kapitel Kantenlokalisierung [▶ 1512] erklärt.

Expert-Parameter

Die Expert-Variante F_VN_LocateEllipse(Exp) [▶ 1541] enthält zusätzliche Parameter.

Anwendung

Die Lokalisierung und Vermessung einer Ellipse mit einem geschätzten Mittelpunkt von `[360, 240]` und einem Radius von `100px` sieht z. B: so aus:

```
VAR
    aGuessedCenter          :   TcVnPoint2_REAL := [360, 240];
    fGuessedRadius          :   REAL := 100;
END_VAR

hr := F_VN_LocateEllipseExp(
    ipSrcImage              :=  ipImageIn,
    stEllipse               :=  stEllipse,
    aCenterPoint            :=  aGuessedCenter,
    fSearchRadius           :=  (fGuessedRadius + 15), // increase search radius to make sure
ellipse is found!
    eEdgeDirection          :=  TCVN_ED_LIGHT_TO_DARK,
    fMinStrength            :=  fMinStrength,
    hrPrev                  :=  hr
);
```

Die Eigenschaften der lokalisierten Ellipse können dann über die Variablen `stEllipse.aCenter`, `stEllipse.stSize` und `stEllipse.fAngle` abgerufen werden. Zudem kann die Ellipse mit der Funktion F_VN_DrawEllipse [▶ 1043] visualisiert werden:

```
hr := F_VN_DrawEllipse(stEllipse, ipImageRes, aColorGreen, 5, hr);
```

Beispiele

- Locate Ellipse [▶ 2909]

Verwandte Funktionen

- LocateCircularArc [▶ 1520] zur Lokalisierung von anteiligen Kreisbögen
- LocateEdge [▶ 1526] zur Lokalisierung von linearen Kanten
- LocateEdges [▶ 1532] zur Lokalisierung von mehreren linearen Kanten
- LocateEllipse [▶ 1538] zur Lokalisierung von vollständigen Ellipsen

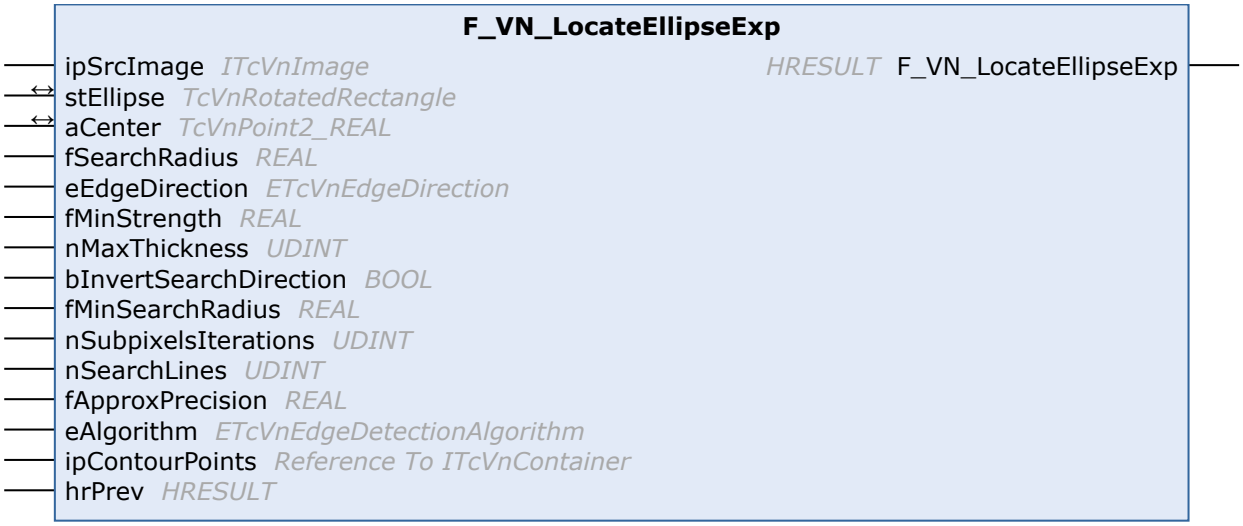
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.20.10 F_VN_LocateEllipseExp



Locate an ellipse. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_LocateEllipseExp : HRESULT
VAR_INPUT
    ipSrcImage             : ITcVnImage;
END_VAR
VAR_IN_OUT
    stEllipse              : TcVnRotatedRectangle;
    aCenter                : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    fSearchRadius          : REAL;
    eEdgeDirection         : ETcVnEdgeDirection;
    fMinStrength           : REAL;
    nMaxThickness          : UDINT;
    bInvertSearchDirection : BOOL;
    fMinSearchRadius       : REAL;
    nSubpixelsIterations   : UDINT;
    nSearchLines           : UDINT;
    fApproxPrecision       : REAL;
    eAlgorithm             : ETcVnEdgeDetectionAlgorithm;
    ipContourPoints        : Reference To ITcVnContainer;
    hrPrev                 : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel)
fSearchRadius	REAL	Radius around aCenter to search for edges
eEdgeDirection	ETcVnEdgeDirection [▶ 210]	Specification of the edge direction to search for (from center to outside ellipse or other way round, if bInvertSearchDirection is true)
fMinStrength	REAL	Specification of the minimum strength (intensity difference) of the edge to search for
nMaxThickness	UDINT	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels
bInvertSearchDirection	BOOL	If true, the search starts from outside the ellipse in direction of the center
fMinSearchRadius	REAL	Radius around aCenter to skip before starting to search for edges (e.g. to save time or if the center contains edges that should be ignored)
nSubpixelsIterations	UDINT	Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)
nSearchLines	UDINT	Specifies the amount of search lines, which are equally distributed in all directions (must be >= 8 and a multiple of 4)
fApproxPrecision	REAL	Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)
eAlgorithm	ETcVnEdgeDetectionAlgorithm [▶ 209]	Selection of the edge detection algorithm
ipContourPoints	Reference To ITcVnContainer [▶ 385]	Returns the subpixel accurate ellipse contour (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_REAL; Non-zero interface pointers are reused.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stEllipse	TcVnRotatedRectangle [▶ 258]	Returns the detected ellipse
aCenter	TcVnPoint2_REAL [▶ 171]	The expected ellipse center

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_LocateEllipseExp` ist die Expert-Variante von F_VN_LocateEllipse [▶ 1538].

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Grauwertbild sein.

Ellipse (Rückgabewert)

Der Parameter stEllipse liefert die lokalisierte Ellipse als Struktur vom Typ TcVnRotatedRectangle [▶ 258] zurück (Mittelpunkt, Größe und Winkel).

Suchkreis

Der kreisförmige Suchbereich wird über den erwarteten Mittelpunkt `aCenter` und sowie einen Suchradius `fSearchRadius` definiert. Der Suchradius sollte etwas größer als der erwartete Ellipsenradius festgelegt werden, um die Ellipse sicher zu lokalisieren.

Lokalisierte Punkte (Expert, Rückgabewert)

Der Container `ipContourPoints` gibt die lokalisierten Punkte zurück.

Wenn die Rückgabe der Kantenpunkte nicht benötigt wird, kann der Parameter auf 0 gesetzt werden.

Parameter zur Kantenlokalisierung

Die restlichen Parameter sind ausführlich im Kapitel Kantenlokalisierung [▶ 1512] erklärt.

Anwendung

Die Lokalisierung und Vermessung einer Ellipse mit Expert-Parametern sieht z. B: so aus:

```
hr := F_VN_LocateEllipseExp(
    ipSrcImage              :=  ipImageIn,
    stEllipse               :=  stEllipse,
    aCenterPoint            :=  aGuessedCenter,
    fSearchRadius           :=  (fGuessedRadius + 15), // increase search radius to make sure
ellipse is found!
    eEdgeDirection          :=  TCVN_ED_LIGHT_TO_DARK,
    fMinStrength            :=  fMinStrength,
    nMaxThickness           :=  7,
    bInvertSearchDirection  :=  TRUE,
    fMinSearchRadius        :=  fMinRadius,
    nSubpixelsIterations    :=  nSubpixIter,
    nSearchLines            :=  nSearchLines,
    fApproxPrecision        :=  0.0001,
    eAlgorithm              :=  eAlgorithm,
    ipContourPoints         :=  ipContourPoints,
    hrPrev                  :=  hr
);
```

Die lokalisierten Punkte der Ellipse können mit der Funktion F_VN_DrawContours [▶ 1040] visualisiert werden:

```
hr := F_VN_DrawContours(ipContourPoints, -1, ipImageRes, aColorGreen, 5, hr);
```

Beispiele

- Locate Ellipse [▶ 2909]

Verwandte Funktionen

- LocateCircularArc [▶ 1520] zur Lokalisierung von anteiligen Kreisbögen
- LocateEdge [▶ 1526] zur Lokalisierung von linearen Kanten
- LocateEdges [▶ 1532] zur Lokalisierung von mehreren linearen Kanten
- LocateEllipse [▶ 1538] zur Lokalisierung von vollständigen Ellipsen

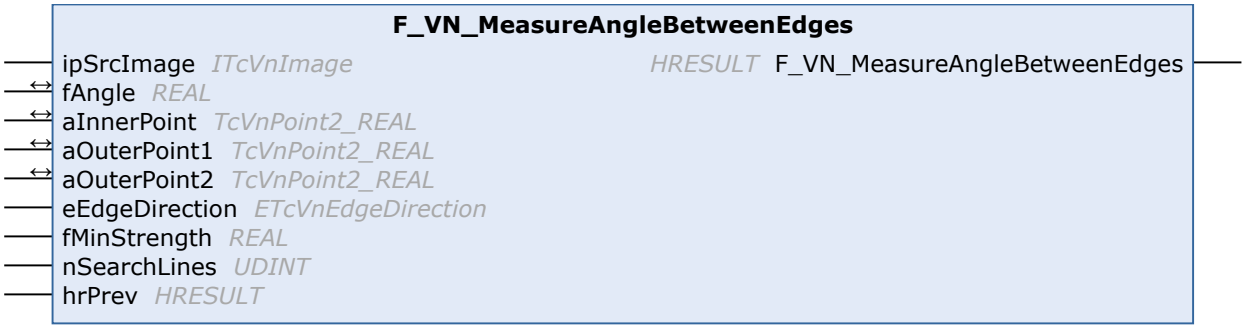
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.20.11 F_VN_MeasureAngleBetweenEdges



Measure the angle between 2 edges, using an interpolation method.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_MeasureAngleBetweenEdges : HRESULT
VAR_INPUT
    ipSrcImage     : ITcVnImage;
END_VAR
VAR_IN_OUT
    fAngle         : REAL;
    aInnerPoint    : TcVnPoint2_REAL;
    aOuterPoint1   : TcVnPoint2_REAL;
    aOuterPoint2   : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    eEdgeDirection : ETcVnEdgeDirection;
    fMinStrength   : REAL;
    nSearchLines   : UDINT;
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel)
eEdgeDirection	ETcVnEdgeDirection [▶ 210]	Specification of the edge direction from aInnerPoint to aOuterPoint1 and aOuterPoint2 to search for
fMinStrength	REAL	Specification of the minimum strength (intensity difference) of the edge to search for
nSearchLines	UDINT	Width of the search windows, i.e. the number of search lines (3, 5, 7, ...), centered around the lines specified by aInnerPoint and aOuterPoint1, aOuterPoint2
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fAngle	REAL	Returns the angle between the detected edges (radians)
aInnerPoint	TcVnPoint2_REAL [▶ 171]	Position inside the angle to measure, from which to start the edge search process (in the direction of aOuterPoint1 and aOuterPoint2)
aOuterPoint1	TcVnPoint2_REAL [▶ 171]	Position where the search process for edge 1 ends
aOuterPoint2	TcVnPoint2_REAL [▶ 171]	Position where the search process for edge 2 ends

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_MeasureAngleBetweenEdges` lokalisiert zwei lineare Kanten und misst den Winkel zwischen ihnen.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Grauwertbild sein.

Winkel (Rückgabewert)

Der Parameter `fAngle` gibt den Winkel zwischen den lokalisierten Kanten zurück.

Abhängig vom Parameter `bAngleInDegrees` ist der Winkel `fAngle` in Radiant (`FALSE`) oder in Grad (`TRUE`) angegeben.

Parameter zur Kantenlokalisierung

Die restlichen Parameter sind ausführlich im Kapitel Kantenlokalisierung [▶ 1512] erklärt.

Expert-Parameter

Die Expert-Variante F_VN_MeasureAngleBetweenEdgesExp [▶ 1546] enthält zusätzliche Parameter.

Anwendung

Das Vermessen eines Winkels zwischen zwei linearen Kanten sieht z. B. so aus:

```
hr := F_VN_MeasureAngleBetweenEdgesExp(
    ipSrcImage              :=   ipImageIn,
    fAngle                  :=   fAngle
    aInnerPoint             :=   aInnerPoint,
    aOuterPoint1            :=   aOuterPoint1
    aOuterPoint2            :=   aOuterPoint2,
    eEdgeDirection          :=   TCVN_ED_LIGHT_TO_DARK,
    fMinStrength            :=   fMinStrength,
    nSearchLines            :=   nSearchLines,
    hrPrev                  :=   hr
);
```

Beispiele

- Measure Angle Between Edges [▶ 2914]

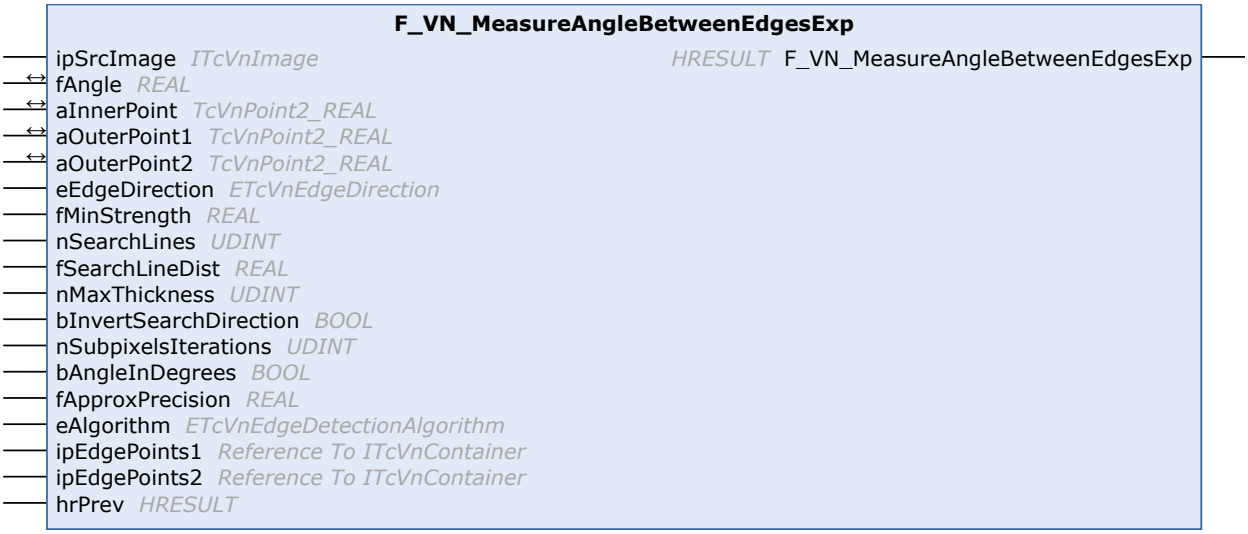
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.20.12 F_VN_MeasureAngleBetweenEdgesExp



Measure the angle between 2 edges.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_MeasureAngleBetweenEdgesExp : HRESULT
VAR_INPUT
    ipSrcImage              : ITcVnImage;
END_VAR
VAR_IN_OUT
    fAngle                  : REAL;
    aInnerPoint             : TcVnPoint2_REAL;
    aOuterPoint1            : TcVnPoint2_REAL;
    aOuterPoint2            : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    eEdgeDirection          : ETcVnEdgeDirection;
    fMinStrength            : REAL;
    nSearchLines            : UDINT;
    fSearchLineDist         : REAL;
    nMaxThickness           : UDINT;
    bInvertSearchDirection  : BOOL;
    nSubpixelsIterations    : UDINT;
    bAngleInDegrees         : BOOL;
    fApproxPrecision        : REAL;
    eAlgorithm              : ETcVnEdgeDetectionAlgorithm;
    ipEdgePoints1           : Reference To ITcVnContainer;
    ipEdgePoints2           : Reference To ITcVnContainer;
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel)
eEdgeDirection	ETcVnEdgeDirection [▶ 210]	Specification of the edge direction from aInnerPoint to aOuterPoint1 and aOuterPoint2 to search for
fMinStrength	REAL	Specification of the minimum strength (intensity difference) of the edge to search for
nSearchLines	UDINT	Width of the search window, i.e. the number of search lines (3, 5, 7, ...), centered around the line specified by aInnerPoint and aOuterPoint1, aOuterPoint2
fSearchLineDist	REAL	Distance between the search lines in pixels (> 0)
nMaxThickness	UDINT	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels
bInvertSearchDirection	BOOL	If true, the search starts from each aOuterPoint in direction of aInnerPoint
nSubpixelsIterations	UDINT	Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)
bAngleInDegrees	BOOL	fAngle is in degrees, if true
fApproxPrecision	REAL	Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)
eAlgorithm	ETcVnEdgeDetectionAlgorithm [▶ 209]	Selection of the edge detection algorithm
ipEdgePoints1	Reference To ITcVnContainer [▶ 385]	Returns the detected edge points between aInnerPoint and aOuterPoint1 (ContainerType_Vector_TcVnPoint2_REAL)
ipEdgePoints2	Reference To ITcVnContainer [▶ 385]	Returns the detected edge points between aInnerPoint and aOuterPoint2 (ContainerType_Vector_TcVnPoint2_REAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fAngle	REAL	Returns the angle between the detected edges (radians or degrees, depending on bAngleInDegrees)
aInnerPoint	TcVnPoint2_REAL [▶ 171]	Position inside the angle to measure, from which to start the edge search process (in the direction of aOuterPoint1 and aOuterPoint2)
aOuterPoint1	TcVnPoint2_REAL [▶ 171]	Position where the search process for edge 1 ends
aOuterPoint2	TcVnPoint2_REAL [▶ 171]	Position where the search process for edge 2 ends

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_MeasureAngleBetweenEdgesExp` ist die Expert-Variante von F_VN_MeasureAngleBetweenEdges [▶ 1544].

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Grauwertbild sein.

Winkel (Rückgabewert)

Der Parameter `fAngle` gibt den Winkel zwischen den lokalisierten Kanten zurück.

Abhängig vom Parameter `bAngleInDegrees` ist der Winkel `fAngle` in Radiant (`FALSE`) oder in Grad (`TRUE`) angegeben.

Parameter zur Kantenlokalisierung

Die restlichen Parameter sind ausführlich im Kapitel Kantenlokalisierung [▶ 1512] erklärt.

Lokalisierte Punkte (Expert, Rückgabewert)

Die Container `ipEdgePoints1` und `ipEdgePoints2` geben die exakten Punkte zurück, die anhand jeder Suchlinie lokalisiert wurden. Falls nicht benötigt, kann anstatt eines Interface Pointers der Wert `0` übergeben werden

Parameter zur Kantenlokalisierung

Die restlichen Parameter sind ausführlich im Kapitel Kantenlokalisierung [▶ 1512] erklärt.

Anwendung

Das Vermessen eines Winkels mit Expert-Parametern sieht z. B. so aus:

```
hr := F_VN_MeasureAngleBetweenEdgesExp(
    ipSrcImage              :=   ipImageIn,
    fAngle                  :=   fAngle
    aInnerPoint             :=   aInnerPoint,
    aOuterPoint1            :=   aOuterPoint1
    aOuterPoint2            :=   aOuterPoint2,
    eEdgeDirection          :=   TCVN_ED_LIGHT_TO_DARK,
    fMinStrength            :=   fMinStrength,
    nSearchLines            :=   nSearchLines,
    fSearchLineDist         :=   fSearchLineDist,
    nMaxThickness           :=   7,
    nSubpixelsIterations    :=   100,
    bAngleInDegrees         :=   TRUE,
    fApproxPrecision        :=   0.0001,
    eAlgorithm              :=   TCVN_EDA_APPROX_ERF,
    ipEdgePoints1           :=   ipEdgePoints1,
    ipEdgePoints2           :=   ipEdgePoints2,
    hrPrev                  :=   hr
);
```

Beispiele

- Measure Angle Between Edges [▶ 2914]

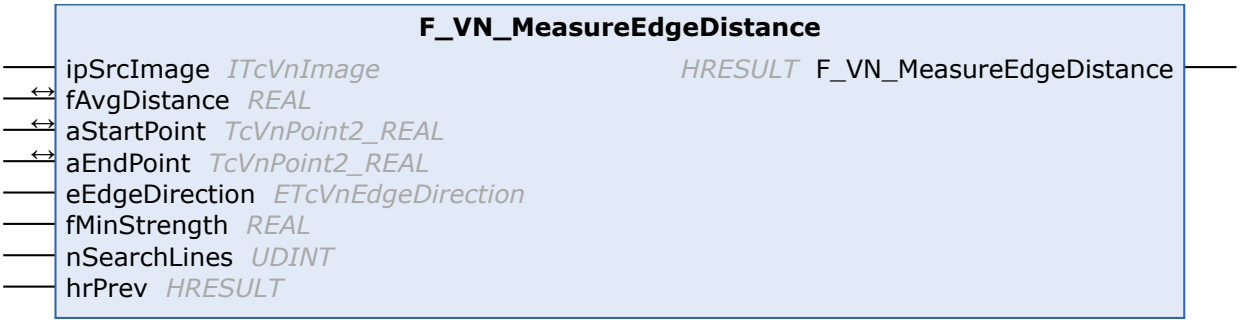
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.20.13 F_VN_MeasureEdgeDistance



Measure the distance between 2 parallel edges, using an interpolation method.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_MeasureEdgeDistance : HRESULT
VAR_INPUT
    ipSrcImage     : ITcVnImage;
END_VAR
VAR_IN_OUT
    fAvgDistance   : REAL;
    aStartPoint    : TcVnPoint2_REAL;
    aEndPoint      : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    eEdgeDirection : ETcVnEdgeDirection;
    fMinStrength   : REAL;
    nSearchLines   : UDINT;
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel)
eEdgeDirection	ETcVnEdgeDirection [▶ 210]	Specification of the edge direction from aStartPoint to aEndPoint to search for
fMinStrength	REAL	Specification of the minimum strength (intensity difference) of the edge to search for
nSearchLines	UDINT	Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ...), centered around the line specified by aStartPoint and aEndPoint
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fAvgDistance	REAL	Returns the average distance between the detected edges
aStartPoint	TcVnPoint2_REAL [▶ 171]	Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 171]	Position where the search process ends

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_MeasureEdgeDistance` lokalisiert zwei lineare Kanten und misst den durchschnittlichen Abstand zwischen ihnen.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Grauwertbild sein.

Minimaler Abstand (Rückgabewert)

Der Parameter `fAvgDistance` gibt des minimalen Abstand der beiden lokalisierten Kanten zurück.

Parameter zur Kantenlokalisierung

Die restlichen Parameter sind ausführlich im Kapitel Kantenlokalisierung [▶ 1512] erklärt.

Expert-Parameter

Die Expert-Variante F_VN_MeasureEdgeDistanceExp [▶ 1551] enthält zusätzliche Parameter.

Anwendung

Die Vermessung einer Kantendistanz sieht z. B. so aus:

```
hr := F_VN_MeasureEdgeDistance(
    ipSrcImage              :=   ipImageIn,
    fAvgDistance            :=   fDistance
    aStartPoint             :=   aStartPoint,
    aEndPoint               :=   aEndPoint,
    eEdgeDirection          :=   TCVN_ED_LIGHT_TO_DARK,
    fMinStrength            :=   fMinStrength,
    nSearchLines            :=   nSearchLines,
    hrPrev                  :=   hr
);
```

Verwandte Funktionen

- ClosestPointsBF [▶ 1518] zur Berechnung der nächstgelegenen Punkte zweier Punktewolken
- MeasureAngleBetweenEdges [▶ 1544] zur Vermessung des Winkels zwischen zwei Kanten
- MeasureEdgeDistance [▶ 1549] zur Vermessung des durchschnittlichen Abstands zweier Kanten
- MeasureMinEdgeDistance [▶ 1554] zur Vermessung des minimalen Abstands zweier Kanten

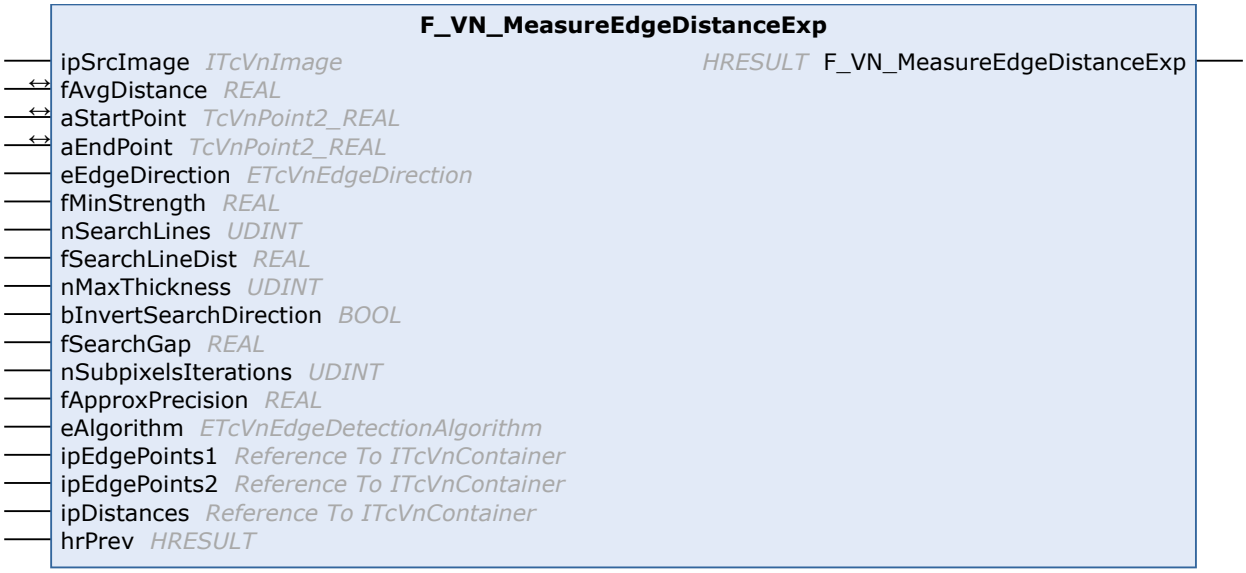
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.20.14 F_VN_MeasureEdgeDistanceExp



Measure the distance between 2 parallel edges. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_MeasureEdgeDistanceExp : HRESULT
VAR_INPUT
    ipSrcImage             : ITcVnImage;
END_VAR
VAR_IN_OUT
    fAvgDistance           : REAL;
    aStartPoint            : TcVnPoint2_REAL;
    aEndPoint              : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    eEdgeDirection         : ETcVnEdgeDirection;
    fMinStrength           : REAL;
    nSearchLines           : UDINT;
    fSearchLineDist        : REAL;
    nMaxThickness          : UDINT;
    bInvertSearchDirection : BOOL;
    fSearchGap             : REAL;
    nSubpixelsIterations   : UDINT;
    fApproxPrecision       : REAL;
    eAlgorithm             : ETcVnEdgeDetectionAlgorithm;
    ipEdgePoints1          : Reference To ITcVnContainer;
    ipEdgePoints2          : Reference To ITcVnContainer;
    ipDistances            : Reference To ITcVnContainer;
    hrPrev                 : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel)
eEdgeDirection	ETcVnEdgeDirection [▶ 210]	Specification of the edge direction from aStartPoint to aEndPoint to search for
fMinStrength	REAL	Specification of the minimum strength (intensity difference) of the edge to search for
nSearchLines	UDINT	Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ...), centered around the line specified by aStartPoint and aEndPoint
fSearchLineDist	REAL	Distance between the search lines in pixels (> 0)
nMaxThickness	UDINT	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels
bInvertSearchDirection	BOOL	If true, the search starts from the center point between aStartPoint and aEndPoint in both directions
fSearchGap	REAL	Optional width of a gap (>= 0, centered between aStartPoint and aEndPoint), that is neglected for searching edges (can reduce execution time). The 2 edges to search for must be on different sides of the gap.
nSubpixelsIterations	UDINT	Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)
fApproxPrecision	REAL	Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)
eAlgorithm	ETcVnEdgeDetectionAlgorithm [▶ 209]	Selection of the edge detection algorithm
ipEdgePoints1	Reference To ITcVnContainer [▶ 385]	Returns the detected edge points of the edge near aStartPoint (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_REAL)
ipEdgePoints2	Reference To ITcVnContainer [▶ 385]	Returns the detected edge points of the edge near aEndPoint (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_REAL)
ipDistances	Reference To ITcVnContainer [▶ 385]	Returns the distances between the detected edge points (optional, set to 0 if not required; ContainerType_Vector_REAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fAvgDistance	REAL	Returns the average distance between the detected edges
aStartPoint	TcVnPoint2_REAL [▶ 171]	Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 171]	Position where the search process ends

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_MeasureEdgeDistanceExp` ist die Expert-Variante von F_VN_MeasureEdgeDistance [▶ 1549]. Sie enthält zusätzliche Parameter.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Grauwertbild sein.

Minimaler Abstand (Rückgabewert)

Der Parameter `fAvgDistance` gibt des minimalen Abstand der beiden lokalisierten Kanten zurück.

Lokalisierte Kantenpunkte (Expert, Rückgabewert)

Die Container `ipEdgePoints1` und `ipEdgePoints2` geben die exakten Punkte zurück, die anhand jeder Suchlinie lokalisiert wurden. Falls nicht benötigt, kann anstatt eines Interface Pointers der Wert `0` übergeben werden.

Gemessene Abstände (Expert, Rückgabewert)

Der Container `ipDistances` gibt alle gemessenen Abstände zwischen gegenüberliegenden Punkten zurück. Falls nicht benötigt, kann anstatt eines Interface Pointers der Wert `0` übergeben werden.

Such-Lücke (Expert)

Bereich in der Mitte der definierten Suchlinie, der bei der Suche ausgelassen wird. Dadurch kann die Ausführungszeit verringert werden.

Parameter zur Kantenlokalisierung

Die restlichen Parameter sind ausführlich im Kapitel Kantenlokalisierung [▶ 1512] erklärt.

Anwendung

Die Vermessung einer Kantendistanz mit Expert-Parametern sieht z. B. so aus:

```
hr := F_VN_MeasureEdgeDistance(
    ipSrcImage              :=   ipImageIn,
    fAvgDistance            :=   fDistance
    aStartPoint             :=   aStartPoint,
    aEndPoint               :=   aEndPoint,
    eEdgeDirection          :=   TCVN_ED_LIGHT_TO_DARK,
    fMinStrength            :=   fMinStrength,
    nSearchLines            :=   nSearchLines,
    fSearchLineDist         :=  3
    nMaxThickness           :=  7,
    bInvertSearchDirection  :=  TRUE,
    nSubpixelsIterations    :=  nSubpixIter,
    fApproxPrecision        :=  0.0001,
    eAlgorithm              :=  eAlgorithm,
    ipEdgePoints1           :=  ipEdge1,
    ipEdgePoints2           :=  ipEdge2,
    ipDistances             :=  ipDistances,
    hrPrev                 :=   hr
);
```

Die lokalisierten Kantenpunkte können mit der Funktion F_VN_DrawContours [▶ 1040] visualisiert werden:

```
hr := F_VN_DrawContours(ipEdgePoints1, -1, ipImageRes, aColorGreen, 5, hr);
hr := F_VN_DrawContours(ipEdgePoints2, -1, ipImageRes, aColorGreen, 5, hr);
```

Verwandte Funktionen

- ClosestPointsBF [▶ 1518] zur Berechnung der nächstgelegenen Punkte zweier Punktewolken
- MeasureAngleBetweenEdges [▶ 1544] zur Vermessung des Winkels zwischen zwei Kanten
- MeasureEdgeDistance [▶ 1549] zur Vermessung des durchschnittlichen Abstands zweier Kanten

- MeasureMinEdgeDistance [▶ 1554] zur Vermessung des minimalen Abstands zweier Kanten

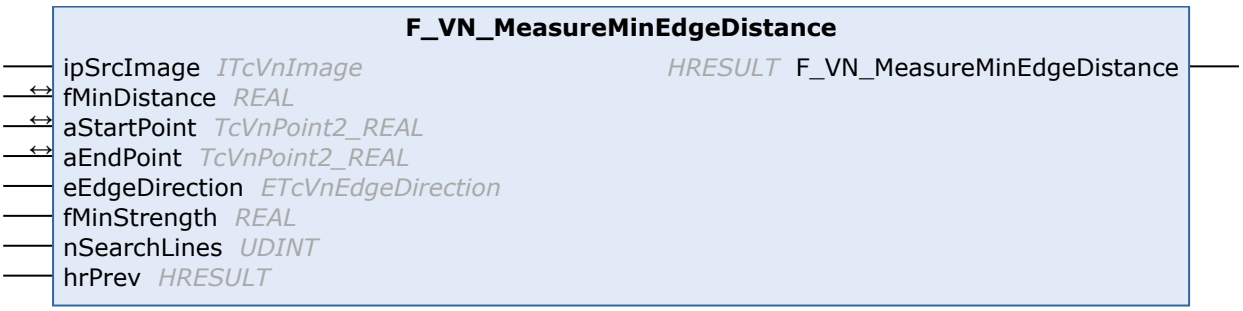
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.20.15 F_VN_MeasureMinEdgeDistance



Measure the minimum distance within the specified search window between 2 edges, using an interpolation method.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_MeasureMinEdgeDistance : HRESULT
VAR_INPUT
    ipSrcImage     : ITcVnImage;
END_VAR
VAR_IN_OUT
    fMinDistance   : REAL;
    aStartPoint    : TcVnPoint2_REAL;
    aEndPoint      : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    eEdgeDirection : ETcVnEdgeDirection;
    fMinStrength   : REAL;
    nSearchLines   : UDINT;
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel)
eEdgeDirection	ETcVnEdgeDirection [▶ 210]	Specification of the edge direction from aStartPoint to aEndPoint to search for
fMinStrength	REAL	Specification of the minimum strength (intensity difference) of the edge to search for
nSearchLines	UDINT	Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ...), centered around the line specified by aStartPoint and aEndPoint
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fMinDistance	REAL	Returns the minimum distance between the detected edges
aStartPoint	TcVnPoint2_REAL [▶ 171]	Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 171]	Position where the search process ends

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_MeasureMinEdgeDistance` lokalisiert zwei lineare Kanten und misst den minimalen Abstand zwischen ihnen. Sie gibt den minimalen Abstand als `REAL` zurück.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Grauwertbild sein.

Minimaler Abstand (Rückgabewert)

Der Parameter `fMinDistance` gibt des minimalen Abstand der beiden lokalisierten Kanten zurück.

Parameter zur Kantenlokalisierung

Die restlichen Parameter sind ausführlich im Kapitel Kantenlokalisierung [▶ 1512] erklärt.

Expert-Parameter

Die Expert-Variante F_VN_MeasureMinEdgeDistanceExp [▶ 1556] enthält zusätzliche Parameter.

Anwendung

Die Vermessung einer Kantendistanz sieht z. B. so aus:

```
hr := F_VN_MeasureMinEdgeDistance(
    ipSrcImage              :=   ipImageIn,
    fMinDistance            :=   fDistance
    aStartPoint             :=   aStartPoint,
    aEndPoint               :=   aEndPoint,
    eEdgeDirection          :=   TCVN_ED_LIGHT_TO_DARK,
    fMinStrength            :=   fMinStrength,
    nSearchLines            :=   nSearchLines,
    hrPrev                  :=   hr
);
```

Verwandte Funktionen

- ClosestPointsBF [▶ 1518] zur Berechnung der nächstgelegenen Punkte zweier Punktewolken
- MeasureAngleBetweenEdges [▶ 1544] zur Vermessung des Winkels zwischen zwei Kanten
- MeasureEdgeDistance [▶ 1549] zur Vermessung des durchschnittlichen Abstands zweier Kanten
- MeasureMinEdgeDistance [▶ 1554] zur Vermessung des minimalen Abstands zweier Kanten

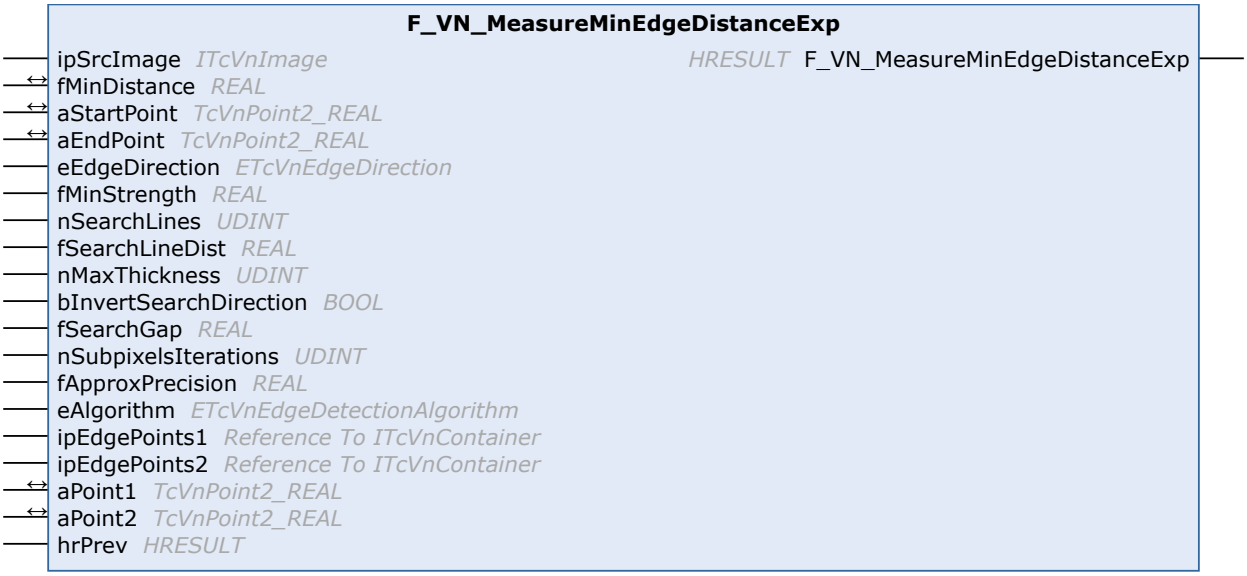
Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.20.16 F_VN_MeasureMinEdgeDistanceExp



Measure the minimum distance within the specified search window between 2 edges. (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
FUNCTION F_VN_MeasureMinEdgeDistanceExp : HRESULT
VAR_INPUT
    ipSrcImage              : ITcVnImage;
END_VAR
VAR_IN_OUT
    fMinDistance            : REAL;
    aStartPoint             : TcVnPoint2_REAL;
    aEndPoint               : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    eEdgeDirection          : ETcVnEdgeDirection;
    fMinStrength            : REAL;
    nSearchLines            : UDINT;
    fSearchLineDist         : REAL;
    nMaxThickness           : UDINT;
    bInvertSearchDirection  : BOOL;
    fSearchGap              : REAL;
    nSubpixelsIterations    : UDINT;
    fApproxPrecision        : REAL;
    eAlgorithm              : ETcVnEdgeDetectionAlgorithm;
    ipEdgePoints1           : Reference To ITcVnContainer;
    ipEdgePoints2           : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    aPoint1                 : TcVnPoint2_REAL;
    aPoint2                 : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    hrPrev                  : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel)
eEdgeDirection	ETcVnEdgeDirection [▶ 210]	Specification of the edge direction from aStartPoint to aEndPoint to search for
fMinStrength	REAL	Specification of the minimum strength (intensity difference) of the edge to search for
nSearchLines	UDINT	Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ...), centered around the line specified by aStartPoint and aEndPoint
fSearchLineDist	REAL	Distance between the search lines in pixels (> 0)
nMaxThickness	UDINT	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels
bInvertSearchDirection	BOOL	If true, the search starts from the center point between aStartPoint and aEndPoint in both directions
fSearchGap	REAL	Optional width of a gap (>= 0, centered between aStartPoint and aEndPoint), that is neglected for searching edges (can reduce execution time). The 2 edges to search for must be on different sides of the gap.
nSubpixelsIterations	UDINT	Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)
fApproxPrecision	REAL	Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)
eAlgorithm	ETcVnEdgeDetectionAlgorithm [▶ 209]	Selection of the edge detection algorithm
ipEdgePoints1	Reference To ITcVnContainer [▶ 385]	Returns the detected edge points of the edge near aStartPoint (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_REAL)
ipEdgePoints2	Reference To ITcVnContainer [▶ 385]	Returns the detected edge points of the edge near aEndPoint (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_REAL)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
fMinDistance	REAL	Returns the minimum distance between the detected edges
aStartPoint	TcVnPoint2_REAL [▶ 171]	Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 171]	Position where the search process ends
aPoint1	TcVnPoint2_REAL [▶ 171]	Returns the point on the edge near aStartPoint, for which the minimum distance is achieved
aPoint2	TcVnPoint2_REAL [▶ 171]	Returns the point on the edge near aEndPoint, for which the minimum distance is achieved

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_MeasureMinEdgeDistanceExp` ist die Expert-Variante von F_VN_MeasureMinEdgeDistance [▶ 1554]. Sie enthält zusätzliche Parameter.

Parameter

Eingangsbild

Das Eingangsbild `ipSrcImage` muss ein 1-kanaliges Grauwertbild sein.

Minimaler Abstand (Rückgabewert)

Der Parameter `fMinDistance` gibt des minimalen Abstand der beiden lokalisierten Kanten zurück.

Lokalisierte Kantenpunkte (Rückgabewert)

Die Container `ipEdgePoints1` und `ipEdgePoints2` geben die exakten Punkte zurück, die anhand jeder Suchlinie lokalisiert wurden. Falls nicht benötigt, kann anstatt eines Interface Pointers der Wert 0 übergeben werden.

Zusätzlich geben die Punkte `aPoint1` und `aPoint2` die Punkte zurück, die den minimalen Abstand zwischen den beiden Kanten haben.

Such-Lücke (Expert)

Bereich in der Mitte der definierten Suchlinie, der bei der Suche ausgelassen wird. Dadurch kann die Ausführungszeit verringert werden.

Parameter zur Kantenlokalisierung

Die restlichen Parameter sind ausführlich im Kapitel Kantenlokalisierung [▶ 1512] erklärt.

Anwendung

Die Vermessung der minimalen Kantendistanz mit Expert-Parametern sieht z. B. so aus:

```
hr := F_VN_MeasureEdgeDistance(
    ipSrcImage              :=  ipImageIn,
    fMinDistance            :=  fDistance
    aStartPoint             :=  aStartPoint,
    aEndPoint               :=  aEndPoint,
    eEdgeDirection          :=  TCVN_ED_LIGHT_TO_DARK,
    fMinStrength            :=  fMinStrength,
    nSearchLines            :=  nSearchLines,
    fSearchLineDist         :=  3
    nMaxThickness           :=  7,
    bInvertSearchDirection  :=  TRUE,
    nSubpixelsIterations    :=  100,
    fApproxPrecision        :=  0.0001,
    eAlgorithm              :=  TCVN_EDA_APPROX_ERF,
    ipEdgePoints1           :=  ipEdge1,
    ipEdgePoints2           :=  ipEdge2,
    aPoint1                 :=  aPoint1,
    aPoint2                 :=  aPoint2,
    hrPrev                  :=  hr
);
```

Die lokalisierten Kantenpunkte können mit den Funktionen F_VN_DrawContours [▶ 1040] und F_VN_DrawPoint [▶ 1062] visualisiert werden:

```
hr := F_VN_DrawContours(ipEdgePoints1, -1, ipImageRes, aColorGreen, 5, hr);
hr := F_VN_DrawContours(ipEdgePoints2, -1, ipImageRes, aColorGreen, 5, hr);

F_VN_DrawPoint(aPoint1[0], aPoint1[1], ipImageRes, TCVN_DS_X, aColorGreen, hr);
F_VN_DrawPoint(aPoint2[0], aPoint2[1], ipImageRes, TCVN_DS_X, aColorGreen, hr);
```

Verwandte Funktionen

- ClosestPointsBF [▶ 1518] zur Berechnung der nächstgelegenen Punkte zweier Punktewolken
- MeasureAngleBetweenEdges [▶ 1544] zur Vermessung des Winkels zwischen zwei Kanten
- MeasureEdgeDistance [▶ 1549] zur Vermessung des durchschnittlichen Abstands zweier Kanten
- MeasureMinEdgeDistance [▶ 1554] zur Vermessung des minimalen Abstands zweier Kanten

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21 Miscellaneous

Diese Gruppe enthält weitere Funktionen, die Gruppenübergreifend verwendet werden können.

Funktionen

Geometrische Hilfsfunktionen

- F_VN_ClipLineToBoundary [▶ 1561]
- F_VN_ClipLineToBoundary_ITcVnImage [▶ 1562]
- F_VN_ClipLineToBoundary_TcVnRectangle_DINT [▶ 1563]
- F_VN_LineIntersectionPoint [▶ 1573]
- F_VN_LineIntersectionPointAndAngle [▶ 1574]
- F_VN_RotatedRectangleCorners [▶ 1587]
- F_VN_RotatedRectangleIntersection [▶ 1587]
- F_VN_OrthogonalVector_TcVnVector2_DINT [▶ 1576]
- F_VN_OrthogonalVector_TcVnVector2_LREAL [▶ 1576]
- F_VN_OrthogonalVector_TcVnVector2_REAL [▶ 1577]
- F_VN_PointToLineDistance_TcVnPoint2_DINT [▶ 1578]
- F_VN_PointToLineDistance_TcVnPoint2_LREAL [▶ 1579]
- F_VN_PointToLineDistance_TcVnPoint2_REAL [▶ 1580]
- F_VN_PointToPointDirection_TcVnPoint2_DINT [▶ 1581]
- F_VN_PointToPointDirection_TcVnPoint2_LREAL [▶ 1582]
- F_VN_PointToPointDirection_TcVnPoint2_REAL [▶ 1583]
- F_VN_PointToPointDistance_TcVnPoint2_DINT [▶ 1584]
- F_VN_PointToPointDistance_TcVnPoint2_LREAL [▶ 1585]
- F_VN_PointToPointDistance_TcVnPoint2_REAL [▶ 1586]

Zeitstempel

- F_VN_GetTimestamp [▶ 1568]
- F_VN_UpdateTimestamp [▶ 1589]

Hilfsfunktionen für Matrizen

- F_VN_InitMatrixStruct [▶ 1570]
- F_VN_InvertMatrix3x3 [▶ 1572]
- F_VN_MultiplyMatrices [▶ 1575]

Konvertieren von Interfaces

- F_VN_ConvertITcUnknownToITcVnBitmapExport [▶ 1564]
- F_VN_ConvertITcUnknownToITcVnContainer [▶ 1565]
- F_VN_ConvertITcUnknownToITcVnImage [▶ 1566]
- F_VN_ConvertITcUnknownToITcVnMlModel [▶ 1567]

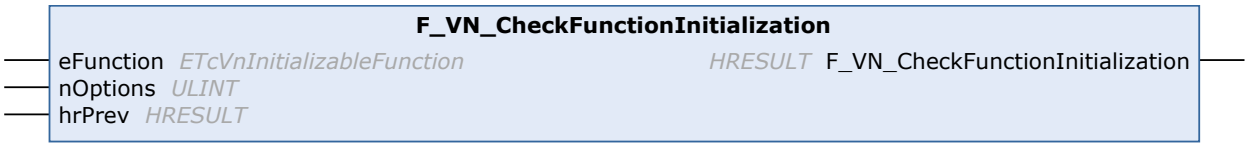
Funktion Initialisierung

- F_VN_CheckFunctionInitialization [▶ 1560]
- F_VN_DeinitializeFunction [▶ 1568]

Sonstiges

- F_VN_HuMomentInvariants [▶ 1569]
- F_VN_SetRngSeed [▶ 1588]

6.1.4.21.1 F_VN_CheckFunctionInitialization



Check if a function is initialized with the specified options.

Syntax

Definition:

```
FUNCTION F_VN_CheckFunctionInitialization : HRESULT
VAR_INPUT
    eFunction : ETcVnInitializableFunction;
    nOptions  : ULINT;
    hrPrev    : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
eFunction	ETcVnInitializableFunction [▶ 217]	Initializable function
nOptions	ULINT	Initialization options for the function
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

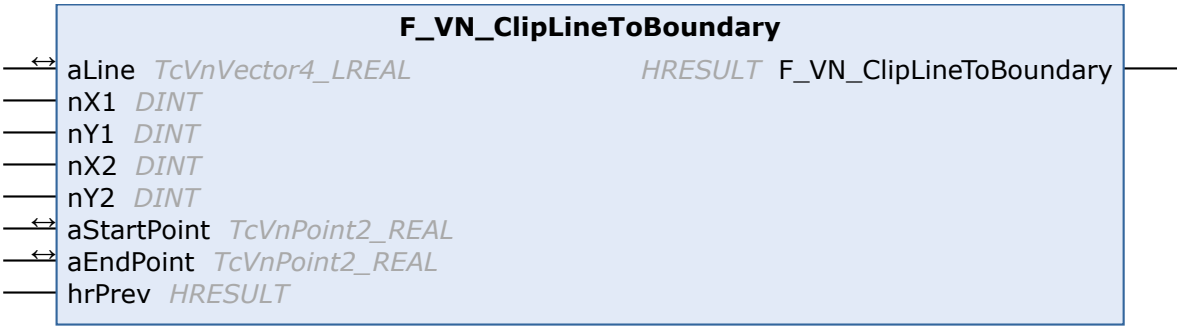
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.2 F_VN_ClipLineToBoundary



Clips a line to a rectangular boundary.

Syntax

Definition:

```
FUNCTION F_VN_ClipLineToBoundary : HRESULT
VAR_IN_OUT
    aLine       : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    nX1         : DINT;
    nY1         : DINT;
    nX2         : DINT;
    nY2         : DINT;
END_VAR
VAR_IN_OUT
    aStartPoint : TcVnPoint2_REAL;
    aEndPoint   : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nX1	DINT	x of top left rectangle point
nY1	DINT	y of top left rectangle point
nX2	DINT	x of bottom right rectangle point
nY2	DINT	Y of bottom right rectangle point
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLine	TcVnVector4_LREAL [▶ 173]	The first and second element describe the x and y component of the direction vector. The third and fourth element describe the x and y component of the position vector.
aStartPoint	TcVnPoint2_REAL [▶ 171]	Returns the clipped starting point of the line.
aEndPoint	TcVnPoint2_REAL [▶ 171]	Returns the clipped end point of the line.

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Diese Funktion passt eine Linie an einen vorgegebenen Bereich an.

Für verschiedene Definitionsformen dieses Bereichs existieren verschiedene Funktionen:

Definition des rechteckigen Bereichs	Zu verwendende Funktion
Durch einzelne X/Y-Koordinaten vorgegeben	F_VN_ClipLineToBoundary [▶ 1561]
Durch die Größe eines Bildes	F_VN_ClipLineToBoundary_ITcVnImage [▶ 1562]
Durch eine Rechteck-Struktur	F_VN_ClipLineToBoundary_TcVnRectangle_DINT [▶ 1563]

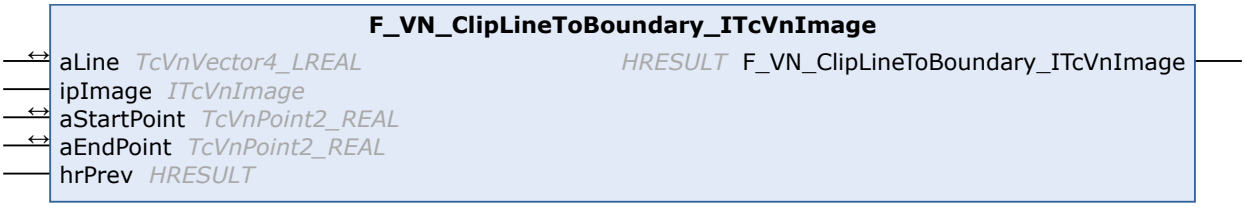
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.3 F_VN_ClipLineToBoundary_ITcVnImage



Clips a line to the boundary of an image.

Syntax

Definition:

```
FUNCTION F_VN_ClipLineToBoundary_ITcVnImage : HRESULT
VAR_IN_OUT
    aLine       : TcVnVector4_LREAL;
END_VAR
VAR_INPUT
    ipImage     : ITcVnImage;
END_VAR
VAR_IN_OUT
    aStartPoint : TcVnPoint2_REAL;
    aEndPoint   : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Image, from which to derive the boundary
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLine	TcVnVector4_LREAL [▶ 173]	The first and second element describe the x and y component of the direction vector. The third and fourth element describe the x and y component of the position vector.
aStartPoint	TcVnPoint2_REAL [▶ 171]	Returns the clipped starting point of the line.
aEndPoint	TcVnPoint2_REAL [▶ 171]	Returns the clipped end point of the line.

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Diese Funktion passt eine Linie an einen vorgegebenen Bereich an.

Für verschiedene Definitionsformen dieses Bereichs existieren verschiedene Funktionen:

Definition des rechteckigen Bereichs	Zu verwendende Funktion
Durch einzelne X/Y-Koordinaten vorgegeben	F_VN_ClipLineToBoundary [▶ 1561]
Durch die Größe eines Bildes	F_VN_ClipLineToBoundary_ITcVnImage [▶ 1562]
Durch eine Rechteck-Struktur	F_VN_ClipLineToBoundary_TcVnRectangle_DINT [▶ 1563]

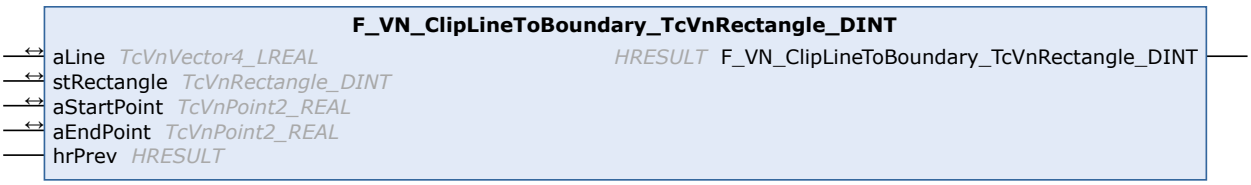
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.4 F_VN_ClipLineToBoundary_TcVnRectangle_DINT



Clips a line to a rectangular boundary.

Syntax

Definition:

```
FUNCTION F_VN_ClipLineToBoundary_TcVnRectangle_DINT : HRESULT
VAR_IN_OUT
    aLine       : TcVnVector4_LREAL;
    stRectangle : TcVnRectangle_DINT;
    aStartPoint : TcVnPoint2_REAL;
    aEndPoint   : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLine	TcVnVector4_LREAL [▶ 173]	The first and second element describe the x and y component of the direction vector. The third and fourth element describe the x and y component of the position vector.
stRectangle	TcVnRectangle_DINT [▶ 257]	Rectangular boundary
aStartPoint	TcVnPoint2_REAL [▶ 171]	Returns the clipped starting point of the line.
aEndPoint	TcVnPoint2_REAL [▶ 171]	Returns the clipped end point of the line.

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Diese Funktion passt eine Linie an einen vorgegebenen Bereich an.

Für verschiedene Definitionsformen dieses Bereichs existieren verschiedene Funktionen:

Definition des rechteckigen Bereichs	Zu verwendende Funktion
Durch einzelne X/Y-Koordinaten vorgegeben	F_VN_ClipLineToBoundary [▶ 1561]
Durch die Größe eines Bildes	F_VN_ClipLineToBoundary_ITcVnImage [▶ 1562]
Durch eine Rechteck-Struktur	F_VN_ClipLineToBoundary_TcVnRectangle_DINT [▶ 1563]

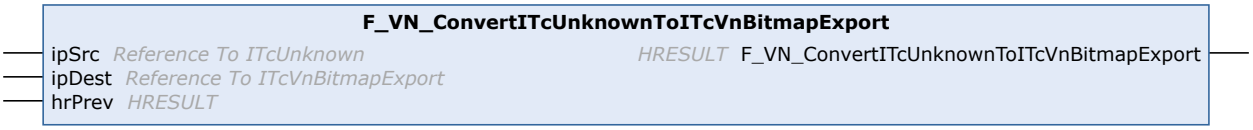
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.5 F_VN_ConvertITcUnknownToITcVnBitmapExport



Convert an ITcUnknown interface pointer to an ITcVnBitmapExport interface pointer. The destination pointer will be released (if existing) and set to the source pointer. The source pointer will be set to 0.

Syntax

Definition:

```
FUNCTION F_VN_ConvertITcUnknownToITcVnBitmapExport : HRESULT
VAR_INPUT
    ipSrc  : Reference To ITcUnknown;
    ipDest : Reference To ITcVnBitmapExport;
    hrPrev : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrc	Reference To ITcUnknown [▶ 443]	Source pointer
ipDest	Reference To ITcVnBitmapExport [▶ 409]	Destination pointer
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

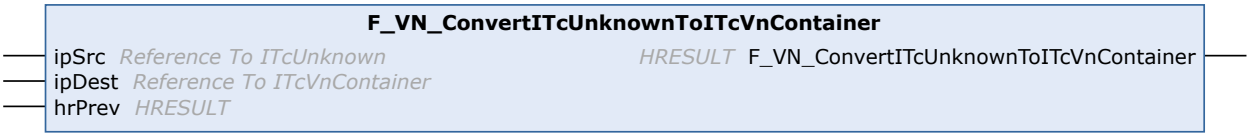
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.6 F_VN_ConvertITcUnknownToITcVnContainer



Convert an ITcUnknown interface pointer to an ITcVnContainer interface pointer. The destination pointer will be released (if existing) and set to the source pointer. The source pointer will be set to 0.

Syntax

Definition:

```
FUNCTION F_VN_ConvertITcUnknownToITcVnContainer : HRESULT
VAR_INPUT
    ipSrc  : Reference To ITcUnknown;
    ipDest : Reference To ITcVnContainer;
    hrPrev : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrc	Reference To ITcUnknown [▶ 443]	Source pointer
ipDest	Reference To ITcVnContainer [▶ 385]	Destination pointer
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

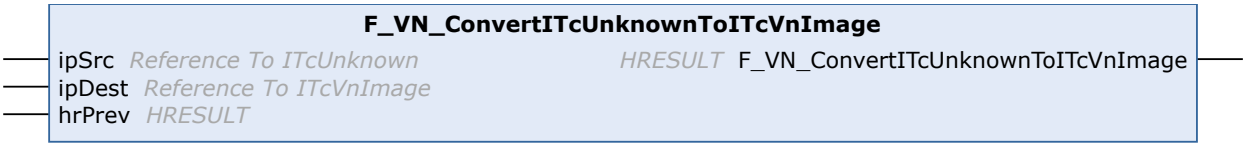
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.7 F_VN_ConvertITcUnknownToITcVnImage



Convert an ITcUnknown interface pointer to an ITcVnImage interface pointer. The destination pointer will be released (if existing) and set to the source pointer. The source pointer will be set to 0.

Syntax

Definition:

```
FUNCTION F_VN_ConvertITcUnknownToITcVnImage : HRESULT
VAR_INPUT
    ipSrc  : Reference To ITcUnknown;
    ipDest : Reference To ITcVnImage;
    hrPrev : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrc	Reference To ITcUnknown [▶ 443]	Source pointer
ipDest	Reference To ITcVnImage [▶ 426]	Destination pointer
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

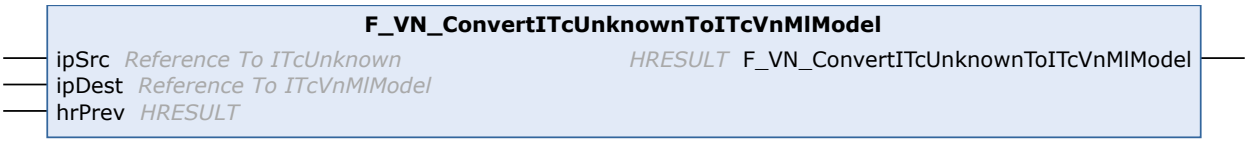
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.8 F_VN_ConvertITcUnknownToITcVnMlModel



Convert an ITcUnknown interface pointer to an ITcVnMlModel interface pointer. The destination pointer will be released (if existing) and set to the source pointer. The source pointer will be set to 0.

Syntax

Definition:

```
FUNCTION F_VN_ConvertITcUnknownToITcVnMlModel : HRESULT
VAR_INPUT
    ipSrc  : Reference To ITcUnknown;
    ipDest : Reference To ITcVnMlModel;
    hrPrev : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrc	Reference To ITcUnknown [▶ 443]	Source pointer
ipDest	Reference To ITcVnMlModel [▶ 435]	Destination pointer
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.9 F_VN_DeinitializeFunction

Deinitialize the specified options for the function.

Syntax

Definition:

```
FUNCTION F_VN_DeinitializeFunction : HRESULT
VAR_INPUT
    eFunction : ETcVnInitializableFunction;
    nOptions  : ULINT;
    hrPrev    : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
eFunction	ETcVnInitializableFunction [▶ 217]	Initializable function
nOptions	ULINT	Initialization options for the function
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

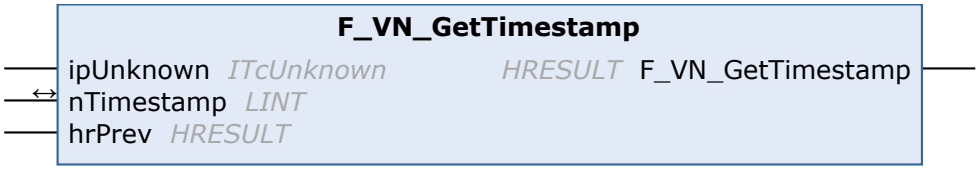
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.10 F_VN_GetTimestamp



Get the timestamp (actual DC time) from the last change of an object (e.g. ITcVnImage or ITcVnContainer)

Syntax

Definition:

```
FUNCTION F_VN_GetTimestamp : HRESULT
VAR_INPUT
    ipUnknown  : ITcUnknown;
END_VAR
VAR_IN_OUT
```

```
    nTimestamp : LINT;
END_VAR
VAR_INPUT
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipUnknown	ITcUnknown [▶ 443]	An object implementing ITcUnknown and ITcVnTimestamp (e.g. ITcVnImage or ITcVnContainer)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
nTimestamp	LINT	Returns the timestamp

Return value

HRESULT [▶ 150]

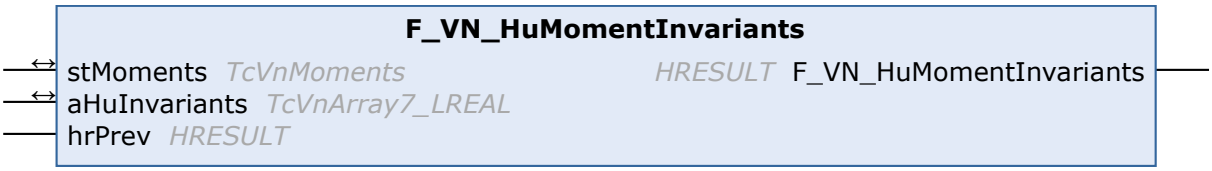
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.11 F_VN_HuMomentInvariants



Computes the Hu moment invariants.

Syntax

Definition:

```
FUNCTION F_VN_HuMomentInvariants : HRESULT
VAR_IN_OUT
    stMoments     : TcVnMoments;
    aHuInvariants : TcVnArray7_LREAL;
END_VAR
VAR_INPUT
    hrPrev        : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stMoments	TcVnMoments [▶ 245]	Moments from which to compute the Hu invariants
aHuInvariants	TcVnArray7_LREAL [▶ 173]	Returns the Hu moment invariants

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.12 F_VN_InitMatrixStruct

Initialize a struct of the type TcVnMatrix extending a buffer with metainformation so that it can be used as a matrix.

Syntax

Definition:

```
FUNCTION F_VN_InitMatrixStruct : HRESULT
VAR_INPUT
    pSrcBuffer   : PVOID;
END_VAR
VAR_IN_OUT
    stDestMatrix : TcVnMatrix;
END_VAR
VAR_INPUT
    nRows        : UDINT;
    nCols        : UDINT;
    eElementType : ETcVnElementType;
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
pSrcBuffer	PVOID	Source buffer
nRows	UDINT	Matrix rows
nCols	UDINT	Matrix columns
eElementType	ETcVnElementType [▶ 210]	Type of the matrix elements
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stDestMatrix	TcVnMatrix [▶ 245]	Returns completed matrix struct

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_InitMatrixStruct` erstellt eine Matrix-Struktur des Typs TcVnMatrix [▶ 245] aus einem beliebigen Speicherbereich.

Parameter

Datenbuffer

Der Pointer `pSrcBuffer` muss auf den Speicherbereich mit den Daten für die Matrix zeigen. Dies kann z. B. ein Array sein.

Matrix-Beschreibung

Die Matrix muss durch die Anzahl der Reihen `nRows` und Spalten `nCols` sowie durch den Datentyp der Element `eElementType` beschrieben werden.

Ergebnis-Matrix

Als Ergebnis wird die Matrix-Struktur `stDestMatrix` vom Typ TcVnMatrix [▶ 245] zurückgegeben.

Anwendung

Die Erstellung einer `3×3`-Matrix sieht z. B. so aus:

```
VAR
    aMatrixArray    :   ARRAY [0..2, 0..2] OF USINT :=
                                            [-1, 0, 1,
                                             -2, 0, 2,
                                             -1, 0, 1];
END_VAR

hr := F_VN_InitMatrixStruct(
    pSrcBuffer      := ADR(aMatrixArray),
    stDestMatrix    := stMatrix,
    nRows           := 3,             // must match with array dimensions
    nCols           := 3,             // must match with array dimensions
    eElementType    := TCVN_ET_USINT, // must match with array type
    hrPrev          := hr);
```

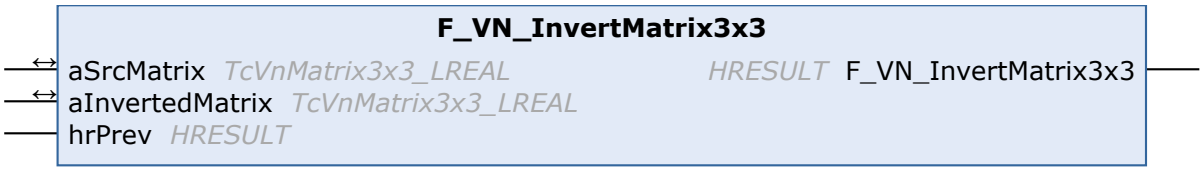
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.13 F_VN_InvertMatrix3x3



Invert a 3x3 matrix.

Syntax

Definition:

```
FUNCTION F_VN_InvertMatrix3x3 : HRESULT
VAR_IN_OUT
    aSrcMatrix      : TcVnMatrix3x3_LREAL;
    aInvertedMatrix : TcVnMatrix3x3_LREAL;
END_VAR
VAR_INPUT
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aSrcMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Source matrix
aInvertedMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns inverted matrix

Return value

HRESULT [▶ 150]

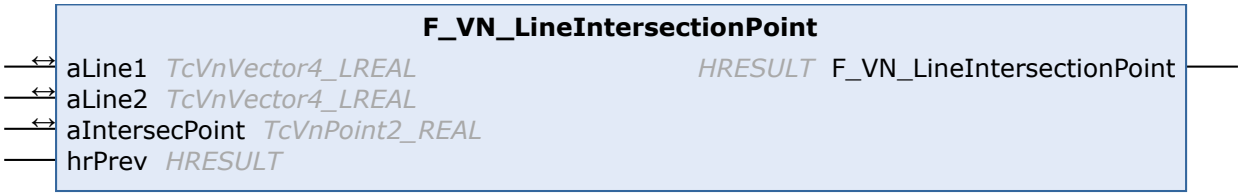
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.14 F_VN_LineIntersectionPoint



Computes the intersection point between two lines (returns S_FALSE if the provided lines are parallel).

Syntax

Definition:

```
FUNCTION F_VN_LineIntersectionPoint : HRESULT
VAR_IN_OUT
    aLine1         : TcVnVector4_LREAL;
    aLine2         : TcVnVector4_LREAL;
    aIntersecPoint : TcVnPoint2_REAL;
END_VAR
VAR_INPUT
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLine1	TcVnVector4_LREAL [▶ 173]	First line. The first and second element describe the x and y component of a vector collinear to the line. The third and fourth element describe the x and y component of a point on the line.
aLine2	TcVnVector4_LREAL [▶ 173]	Second line. The first and second element describe the x and y component of a vector collinear to the line. The third and fourth element describe the x and y component of a point on the line.
aIntersecPoint	TcVnPoint2_REAL [▶ 171]	Returns the intersection point of aLine1 and aLine2.

Return value

HRESULT [▶ 150]

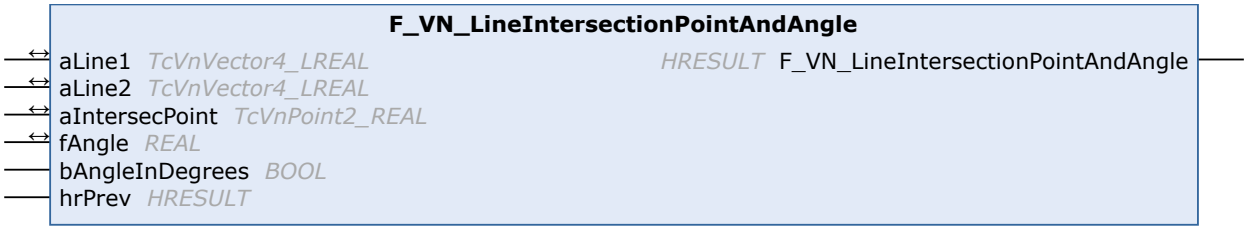
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.15 F_VN_LineIntersectionPointAndAngle



Computes the intersection point and angle between two lines (returns S_FALSE if the provided lines are parallel).

Syntax

Definition:

```
FUNCTION F_VN_LineIntersectionPointAndAngle : HRESULT
VAR_IN_OUT
    aLine1          : TcVnVector4_LREAL;
    aLine2          : TcVnVector4_LREAL;
    aIntersecPoint  : TcVnPoint2_REAL;
    fAngle          : REAL;
END_VAR
VAR_INPUT
    bAngleInDegrees : BOOL;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
bAngleInDegrees	BOOL	If TRUE, fAngle is in degrees. If FALSE, fAngle is in radians.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aLine1	TcVnVector4_LREAL [▶ 173]	First line. The first and second element describe the x and y component of a vector collinear to the line. The third and fourth element describe the x and y component of a point on the line.
aLine2	TcVnVector4_LREAL [▶ 173]	Second line. The first and second element describe the x and y component of a vector collinear to the line. The third and fourth element describe the x and y component of a point on the line.
aIntersecPoint	TcVnPoint2_REAL [▶ 171]	Returns the intersection point of aLine1 and aLine2.
fAngle	REAL	Returns the intersection angle of aLine1 and aLine2.

Return value

HRESULT [▶ 150]

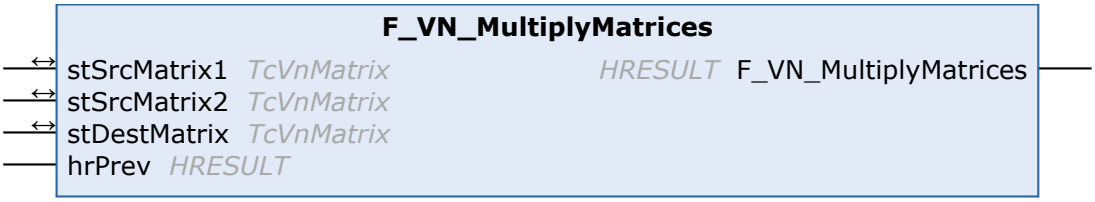
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.16 F_VN_MultiplyMatrices



Matrix multiplication of two matrices (A * B = C). All matrices are represented by structs holding a pointer to an array of the appropriate size.

Syntax

Definition:

```
FUNCTION F_VN_MultiplyMatrices : HRESULT
VAR_IN_OUT
    stSrcMatrix1 : TcVnMatrix;
    stSrcMatrix2 : TcVnMatrix;
    stDestMatrix : TcVnMatrix;
END_VAR
VAR_INPUT
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stSrcMatrix1	TcVnMatrix [▶ 245]	First source matrix (A)
stSrcMatrix2	TcVnMatrix [▶ 245]	Second source matrix (B)
stDestMatrix	TcVnMatrix [▶ 245]	Destination matrix (C) (The destination matrix is filled by this function, but the required memory needs to be provided.)

Return value

HRESULT [▶ 150]

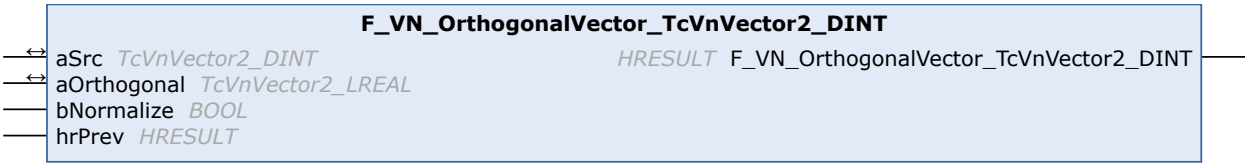
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.17 F_VN_OrthogonalVector_TcVnVector2_DINT



Computes an orthogonal vector to a given source vector.

Syntax

Definition:

```
FUNCTION F_VN_OrthogonalVector_TcVnVector2_DINT : HRESULT
VAR_IN_OUT
    aSrc         : TcVnVector2_DINT;
    aOrthogonal : TcVnVector2_LREAL;
END_VAR
VAR_INPUT
    bNormalize   : BOOL;
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
bNormalize	BOOL	If true, aOrthogonal length is normalized to 1, otherwise aOrthogonal has the same length as aSrc
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aSrc	TcVnVector2_DINT [▶ 173]	Source vector
aOrthogonal	TcVnVector2_LREAL [▶ 173]	Returns a vector orthogonal to aSrc

Return value

HRESULT [▶ 150]

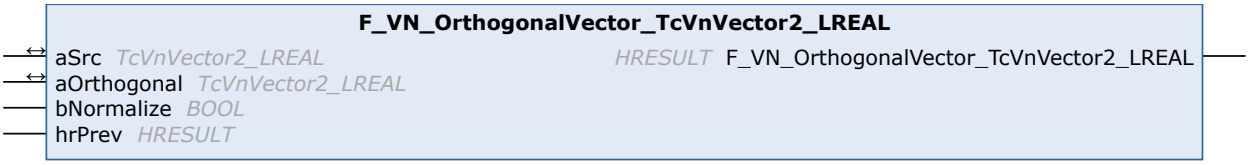
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.18 F_VN_OrthogonalVector_TcVnVector2_LREAL



Computes an orthogonal vector to a given source vector.

Syntax

Definition:

```
FUNCTION F_VN_OrthogonalVector_TcVnVector2_LREAL : HRESULT
VAR_IN_OUT
    aSrc        : TcVnVector2_LREAL;
    aOrthogonal : TcVnVector2_LREAL;
END_VAR
VAR_INPUT
    bNormalize  : BOOL;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
bNormalize	BOOL	If true, aOrthogonal length is normalized to 1, otherwise aOrthogonal has the same length as aSrc
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aSrc	TcVnVector2_LREAL [▶ 173]	Source vector
aOrthogonal	TcVnVector2_LREAL [▶ 173]	Returns a vector orthogonal to aSrc

Return value

HRESULT [▶ 150]

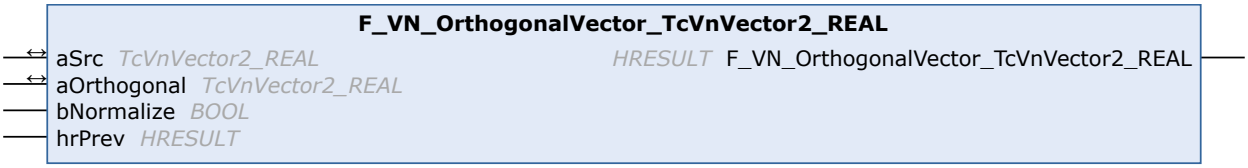
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.19 F_VN_OrthogonalVector_TcVnVector2_REAL



Computes an orthogonal vector to a given source vector.

Syntax

Definition:

```
FUNCTION F_VN_OrthogonalVector_TcVnVector2_REAL : HRESULT
VAR_IN_OUT
    aSrc        : TcVnVector2_REAL;
    aOrthogonal : TcVnVector2_REAL;
END_VAR
VAR_INPUT
    bNormalize  : BOOL;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
bNormalize	BOOL	If true, aOrthogonal length is normalized to 1, otherwise aOrthogonal has the same length as aSrc
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aSrc	TcVnVector2_REAL [▶ 173]	Source vector
aOrthogonal	TcVnVector2_REAL [▶ 173]	Returns a vector orthogonal to aSrc

Return value

HRESULT [▶ 150]

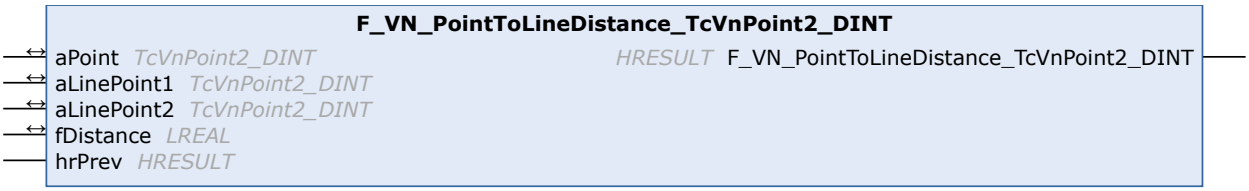
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.20 F_VN_PointToLineDistance_TcVnPoint2_DINT



Computes the shortest distance between a point and a line.

Syntax

Definition:

```
FUNCTION F_VN_PointToLineDistance_TcVnPoint2_DINT : HRESULT
VAR_IN_OUT
    aPoint      : TcVnPoint2_DINT;
    aLinePoint1 : TcVnPoint2_DINT;
    aLinePoint2 : TcVnPoint2_DINT;
    fDistance   : LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aPoint	TcVnPoint2_DINT [▶ 171]	Point
aLinePoint1	TcVnPoint2_DINT [▶ 171]	Line Point 1
aLinePoint2	TcVnPoint2_DINT [▶ 171]	Line Point 2
fDistance	LREAL	Returns the distance between the point and the line

Return value

HRESULT [▶ 150]

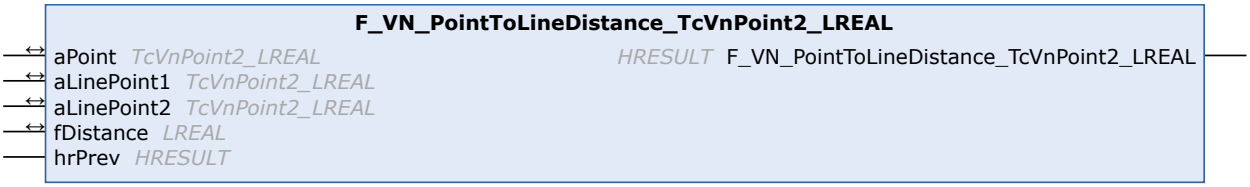
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.21 F_VN_PointToLineDistance_TcVnPoint2_LREAL



Computes the shortest distance between a point and a line.

Syntax

Definition:

```
FUNCTION F_VN_PointToLineDistance_TcVnPoint2_LREAL : HRESULT
VAR_IN_OUT
    aPoint      : TcVnPoint2_LREAL;
    aLinePoint1 : TcVnPoint2_LREAL;
    aLinePoint2 : TcVnPoint2_LREAL;
    fDistance   : LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aPoint	TcVnPoint2_LREAL [▶ 171]	Point
aLinePoint1	TcVnPoint2_LREAL [▶ 171]	Line Point 1
aLinePoint2	TcVnPoint2_LREAL [▶ 171]	Line Point 2
fDistance	LREAL	Returns the distance between the point and the line

Return value

HRESULT [▶ 150]

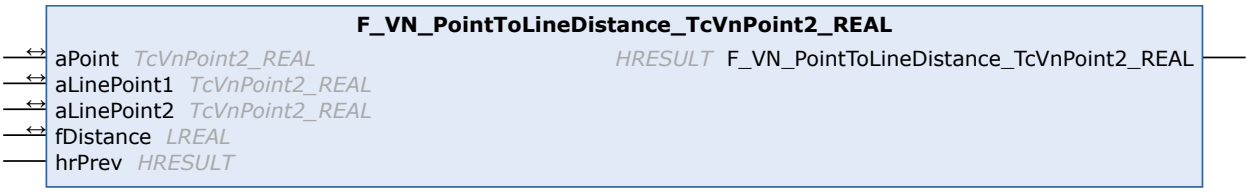
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.22 F_VN_PointToLineDistance_TcVnPoint2_REAL



Computes the shortest distance between a point and a line.

Syntax

Definition:

```
FUNCTION F_VN_PointToLineDistance_TcVnPoint2_REAL : HRESULT
VAR_IN_OUT
    aPoint      : TcVnPoint2_REAL;
    aLinePoint1 : TcVnPoint2_REAL;
    aLinePoint2 : TcVnPoint2_REAL;
    fDistance   : LREAL;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

/ In/Outputs

Name	Type	Description
aPoint	TcVnPoint2_REAL [▶ 171]	Point
aLinePoint1	TcVnPoint2_REAL [▶ 171]	Line Point 1
aLinePoint2	TcVnPoint2_REAL [▶ 171]	Line Point 2
fDistance	LREAL	Returns the distance between the point and the line

Return value

HRESULT [▶ 150]

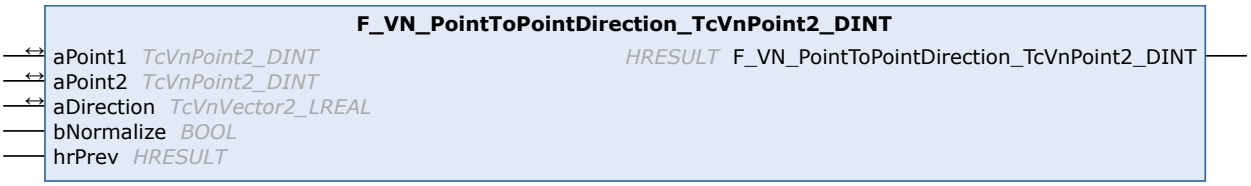
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.23 F_VN_PointToPointDirection_TcVnPoint2_DINT



Computes the direction vector between 2 points.

Syntax

Definition:

```
FUNCTION F_VN_PointToPointDirection_TcVnPoint2_DINT : HRESULT
VAR_IN_OUT
    aPoint1    : TcVnPoint2_DINT;
    aPoint2    : TcVnPoint2_DINT;
    aDirection : TcVnVector2_LREAL;
END_VAR
VAR_INPUT
    bNormalize : BOOL;
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
bNormalize	BOOL	If true, aDirection length is normalized to 1, otherwise aDirection length is the distance between aPoint1 and aPoint2
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

/ In/Outputs

Name	Type	Description
aPoint1	TcVnPoint2_DINT [▶ 171]	Point 1
aPoint2	TcVnPoint2_DINT [▶ 171]	Point 2
aDirection	TcVnVector2_LREAL [▶ 173]	Returns the direction vector from aPoint1 to aPoint2

Return value

HRESULT [▶ 150]

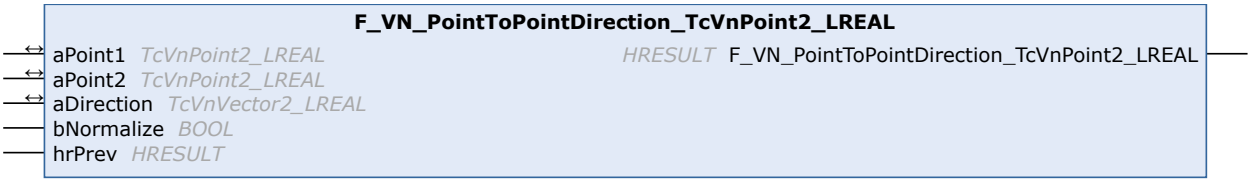
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.24 F_VN_PointToPointDirection_TcVnPoint2_LREAL



Computes the direction vector between 2 points.

Syntax

Definition:

```
FUNCTION F_VN_PointToPointDirection_TcVnPoint2_LREAL : HRESULT
VAR_IN_OUT
    aPoint1    : TcVnPoint2_LREAL;
    aPoint2    : TcVnPoint2_LREAL;
    aDirection : TcVnVector2_LREAL;
END_VAR
VAR_INPUT
    bNormalize : BOOL;
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
bNormalize	BOOL	If true, aDirection length is normalized to 1, otherwise aDirection length is the distance between aPoint1 and aPoint2
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

/ In/Outputs

Name	Type	Description
aPoint1	TcVnPoint2_LREAL [▶ 171]	Point 1
aPoint2	TcVnPoint2_LREAL [▶ 171]	Point 2
aDirection	TcVnVector2_LREAL [▶ 173]	Returns the direction vector from aPoint1 to aPoint2

Return value

HRESULT [▶ 150]

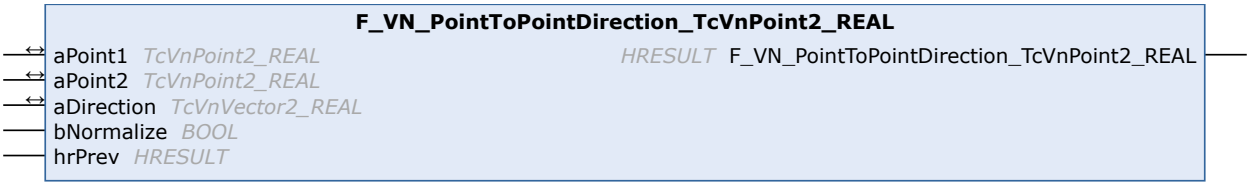
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.25 F_VN_PointToPointDirection_TcVnPoint2_REAL



Computes the direction vector between 2 points.

Syntax

Definition:

```
FUNCTION F_VN_PointToPointDirection_TcVnPoint2_REAL : HRESULT
VAR_IN_OUT
    aPoint1    : TcVnPoint2_REAL;
    aPoint2    : TcVnPoint2_REAL;
    aDirection : TcVnVector2_REAL;
END_VAR
VAR_INPUT
    bNormalize : BOOL;
    hrPrev     : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
bNormalize	BOOL	If true, aDirection length is normalized to 1, otherwise aDirection length is the distance between aPoint1 and aPoint2
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

/ In/Outputs

Name	Type	Description
aPoint1	TcVnPoint2_REAL [▶ 171]	Point 1
aPoint2	TcVnPoint2_REAL [▶ 171]	Point 2
aDirection	TcVnVector2_REAL [▶ 173]	Returns the direction vector from aPoint1 to aPoint2

Return value

HRESULT [▶ 150]

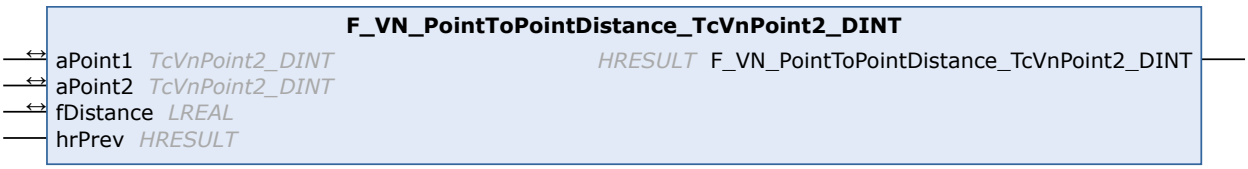
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.26 F_VN_PointToPointDistance_TcVnPoint2_DINT



Computes the distance between 2 points.

Syntax

Definition:

```
FUNCTION F_VN_PointToPointDistance_TcVnPoint2_DINT : HRESULT
VAR_IN_OUT
    aPoint1   : TcVnPoint2_DINT;
    aPoint2   : TcVnPoint2_DINT;
    fDistance : LREAL;
END_VAR
VAR_INPUT
    hrPrev    : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

/ In/Outputs

Name	Type	Description
aPoint1	TcVnPoint2_DINT [▶ 171]	Point 1
aPoint2	TcVnPoint2_DINT [▶ 171]	Point 2
fDistance	LREAL	Returns the distance between the points

Return value

HRESULT [▶ 150]

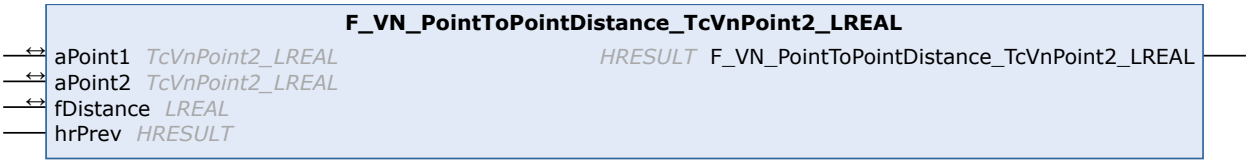
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.27 F_VN_PointToPointDistance_TcVnPoint2_LREAL



Computes the distance between 2 points.

Syntax

Definition:

```
FUNCTION F_VN_PointToPointDistance_TcVnPoint2_LREAL : HRESULT
VAR_IN_OUT
    aPoint1   : TcVnPoint2_LREAL;
    aPoint2   : TcVnPoint2_LREAL;
    fDistance : LREAL;
END_VAR
VAR_INPUT
    hrPrev    : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aPoint1	TcVnPoint2_LREAL [▶ 171]	Point 1
aPoint2	TcVnPoint2_LREAL [▶ 171]	Point 2
fDistance	LREAL	Returns the distance between the points

Return value

HRESULT [▶ 150]

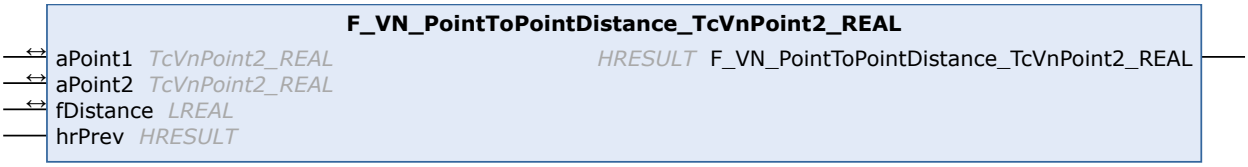
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.28 F_VN_PointToPointDistance_TcVnPoint2_REAL



Computes the distance between 2 points.

Syntax

Definition:

```
FUNCTION F_VN_PointToPointDistance_TcVnPoint2_REAL : HRESULT
VAR_IN_OUT
    aPoint1   : TcVnPoint2_REAL;
    aPoint2   : TcVnPoint2_REAL;
    fDistance : LREAL;
END_VAR
VAR_INPUT
    hrPrev    : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aPoint1	TcVnPoint2_REAL [▶ 171]	Point 1
aPoint2	TcVnPoint2_REAL [▶ 171]	Point 2
fDistance	LREAL	Returns the distance between the points

Return value

HRESULT [▶ 150]

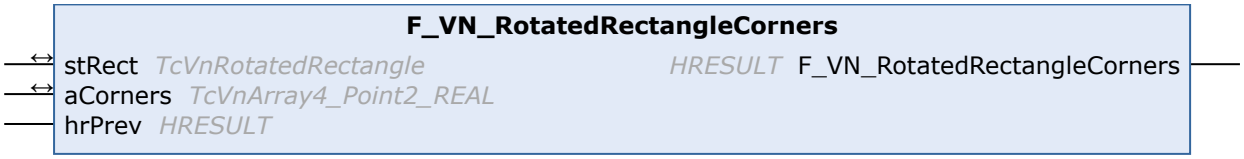
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.29 F_VN_RotatedRectangleCorners



Computes the 4 corner points of a rotated rectangle.

Syntax

Definition:

```
FUNCTION F_VN_RotatedRectangleCorners : HRESULT
VAR_IN_OUT
    stRect   : TcVnRotatedRectangle;
    aCorners : TcVnArray4_Point2_REAL;
END_VAR
VAR_INPUT
    hrPrev   : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRect	TcVnRotatedRectangle [▶ 258]	Rotated rectangle
aCorners	TcVnArray4_Point2_REAL [▶ 173]	Returns the 4 corner points

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.30 F_VN_RotatedRectangleIntersection

Finds the intersection of 2 rotated rectangles.

Syntax

Definition:

```
FUNCTION F_VN_RotatedRectangleIntersection : HRESULT
VAR_IN_OUT
    stRect1        : TcVnRotatedRectangle;
    stRect2        : TcVnRotatedRectangle;
END_VAR
VAR_INPUT
    ipIntersection : Reference To ITcVnContainer;
END_VAR
VAR_IN_OUT
    eIntersection  : ETcVnRectangleIntersection;
END_VAR
VAR_INPUT
    hrPrev         : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipIntersection	Reference To ITcVnContainer [▶ 385]	Returns the intersection points (ContainerType_Vector_TcVnPoint2_REAL, non-zero interface pointers are reused)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
stRect1	TcVnRotatedRectangle [▶ 258]	Rotated rectangle 1
stRect2	TcVnRotatedRectangle [▶ 258]	Rotated rectangle 2
eIntersection	ETcVnRectangleIntersection [▶ 226]	Returns the intersection type

Return value

HRESULT [▶ 150]

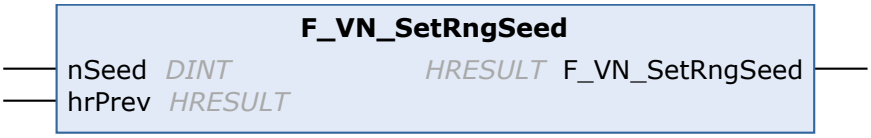
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.31 F_VN_SetRngSeed



Sets the internal pseudo random number generator seed (intended for testing purposes only).

Syntax

Definition:

```
FUNCTION F_VN_SetRngSeed : HRESULT
VAR_INPUT
    nSeed  : DINT;
    hrPrev : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
nSeed	DINT	seed (0 sets the generator back to its initial state)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

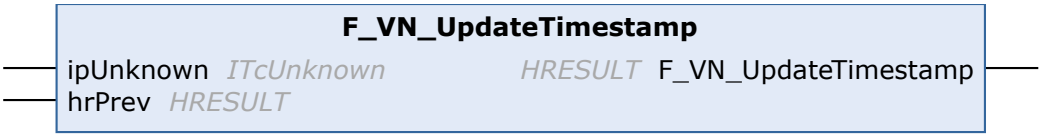
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.21.32 F_VN_UpdateTimestamp



Update the timestamp of an object to the actual DC time (e.g. ITcVnImage or ITcVnContainer)

Syntax

Definition:

```
FUNCTION F_VN_UpdateTimestamp : HRESULT
VAR_INPUT
    ipUnknown : ITcUnknown;
    hrPrev    : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipUnknown	ITcUnknown [▶ 443]	An object implementing ITcUnknown and ITcVnTimestamp (e.g. ITcVnImage or ITcVnContainer)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.22 Neural Network

Diese Gruppe enthält Funktionen zum Pre- und Postprocessing, sowie zur Ausführung von Neural Network Modellen.

Funktionen

Pre- & Postprocessing

- F_VN_ConvertDataLayout [▶ 1591]
- F_VN_ReshapeImage [▶ 1606]

Ausführung von Neuronalen Netzen

- F_VN_ExecuteNeuralNetwork [▶ 1592]
  Mit einem Eingang und einem Ausgang.
- F_VN_ExecuteNeuralNetwork_MI [▶ 1593]
  Mit mehreren Eingängen (MI) und einem Ausgang.
- F_VN_ExecuteNeuralNetwork_MIMO_String [▶ 1594]
  Mit mehreren Eingängen und Ausgängen (MIMO).
- F_VN_ExecuteNeuralNetwork_MISO [▶ 1595]
  Mit mehreren Eingängen und einem einzigen spezifizierten Ausgang (MISO).
- F_VN_ExecuteNeuralNetwork_MO [▶ 1596]
  Ein Eingang und mehrere Ausgänge (MO).
- F_VN_ExecuteNeuralNetwork_SO [▶ 1599]
  Mit einem Eingang und einem einzigen spezifizierten Ausgang (SO).

Postprocessing

- F_VN_Sigmoid(Exp) [▶ 1607]
- F_VN_SoftMax [▶ 1609]
- F_VN_NonMaxSuppression(Exp) [▶ 1602]
- F_VN_Reduce [▶ 1604]
- F_VN_ReduceArg [▶ 1605]

Abfrage der Layer-Bezeichnungen

- F_VN_GetNeuralNetworkLayerNames [▶ 1600]
- F_VN_GetNeuralNetworkLayerNamesExp_String [▶ 1601]

6.1.4.22.1 F_VN_ConvertDataLayout

Convert a data layout (the arrangement of dimensions in the input data) to another data layout.

Syntax

Definition:

```
FUNCTION F_VN_ConvertDataLayout : HRESULT
VAR_INPUT
    ipSrcImage      : ITcVnImage;
    ipDestImage     : Reference To ITcVnImage;
    nSrcDataLayout  : ULINT;
    nDestDataLayout : ULINT;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
nSrcDataLayout	ULINT	Data layout of the source image (Commonly used data layouts are available in the enumeration ETcVnDataLayout. If nSrcDataLayout is set to 0 or TCVN_DL_DEFAULT, the DataLayout will be automatically determined based on the dimension of the ipSrcImage: 0x12 for 2D, 0x123 for 3D, 0x1234 for 4D, and so forth.)
nDestDataLayout	ULINT	Required output data layout (Commonly used data layouts are available in the enumeration ETcVnDataLayout. nDestDataLayout must not equal 0 or TCVN_DL_DEFAULT.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

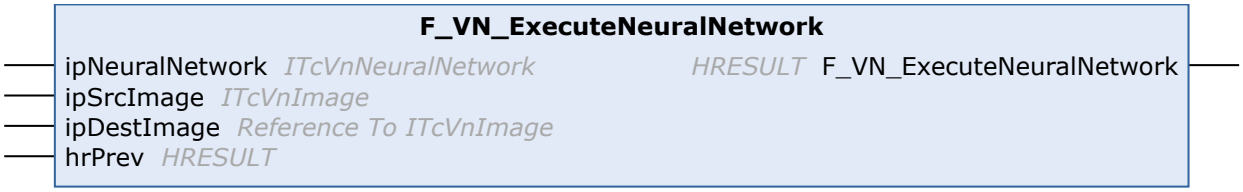
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.22.2 F_VN_ExecuteNeuralNetwork



Execute a neural network using the provided image as input and provide the output of one output layer. The function doesn't perform any pre-processing or transformation.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_ExecuteNeuralNetwork : HRESULT
VAR_INPUT
    ipNeuralNetwork : ITcVnNeuralNetwork;
    ipSrcImage      : ITcVnImage;
    ipDestImage     : Reference To ITcVnImage;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipNeuralNetwork	ITcVnNeuralNetwork [▶ 445]	Neural network model
ipSrcImage	ITcVnImage [▶ 426]	Source image (The image layout must be adapted to match the model's expected input format using e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage functions if necessary.)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image based on the output layer is created. To process the destination image, use e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage if necessary.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

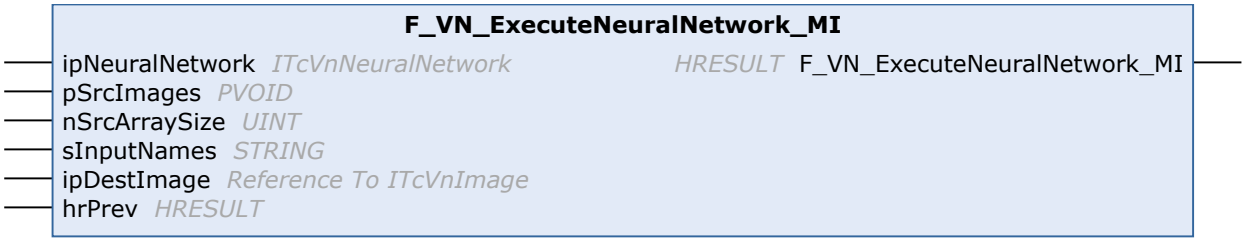
Required License

TC3 Neural Network Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.22.3 F_VN_ExecuteNeuralNetwork_MI



Execute a neural network with multiple inputs (MI) using the provided images. Ensure all model input names are listed in sInputNames and that the images and names are in the same order. The result of one output layer is returned. The function doesn't perform any pre-processing or transformation.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_ExecuteNeuralNetwork_MI : HRESULT
VAR_INPUT
    ipNeuralNetwork : ITcVnNeuralNetwork;
    pSrcImages      : PVOID;
    nSrcArraySize   : UINT;
    sInputNames     : STRING;
    ipDestImage     : Reference To ITcVnImage;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipNeuralNetwork	ITcVnNeuralNetwork [▶ 445]	Neural network model
pSrcImages	PVOID	Pointer to an array of source input images
nSrcArraySize	UINT	The size of the array referenced by pSrcImages. pSrcImages must reference an array with a size equal to the number of model inputs.
sInputNames	STRING	Input names separated by ";"
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image based on the output layer is created. To process the destination image, use e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage if necessary.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

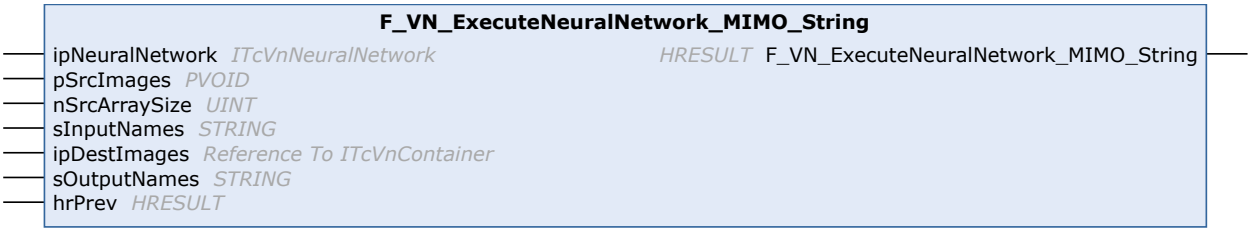
Required License

TC3 Neural Network Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.22.4 F_VN_ExecuteNeuralNetwork_MIMO_String



Execute a neural network with multiple inputs and multiple outputs (MIMO) using the provided images. Ensure all model input names are listed in sInputNames and that the images and names are in the same order. The result of the specified output layers are returned. The function doesn't perform any pre-processing or transformation.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_ExecuteNeuralNetwork_MIMO_String : HRESULT
VAR_INPUT
    ipNeuralNetwork : ITcVnNeuralNetwork;
    pSrcImages      : PVOID;
    nSrcArraySize   : UINT;
    sInputNames     : STRING;
    ipDestImages    : Reference To ITcVnContainer;
    sOutputNames    : STRING;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipNeuralNetwork	ITcVnNeuralNetwork [▶ 445]	Neural network model
pSrcImages	PVOID	Pointer to an array of source input images
nSrcArraySize	UINT	The size of the array referenced by pSrcImages. pSrcImages must reference an array with a size equal to the number of model inputs.
sInputNames	STRING	Input names separated by ";"
ipDestImages	Reference To ITcVnContainer [▶ 385]	Container of images containing the output from the specified layers.
sOutputNames	STRING	Names of output layers whose outputs are required separated by ";". If empty, the outputs of all unconnected output layers are provided.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

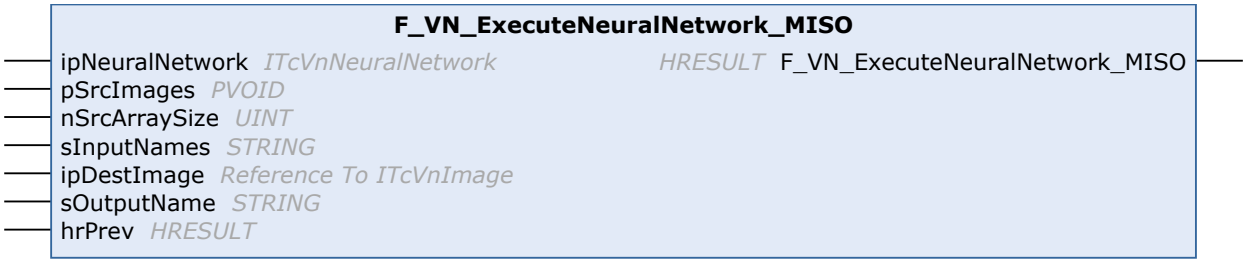
Required License

TC3 Neural Network Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.22.5 F_VN_ExecuteNeuralNetwork_MISO



Execute a neural network with multiple inputs and single output (MISO) using the provided images. Ensure all model input names are listed in sInputNames and that the images and names are in the same order. The result of the specified output layer is returned. The function doesn't perform any pre-processing or transformation.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_ExecuteNeuralNetwork_MISO : HRESULT
VAR_INPUT
    ipNeuralNetwork : ITcVnNeuralNetwork;
    pSrcImages      : PVOID;
    nSrcArraySize   : UINT;
    sInputNames     : STRING;
    ipDestImage     : Reference To ITcVnImage;
    sOutputName     : STRING;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipNeuralNetwork	ITcVnNeuralNetwork [▶ 445]	Neural network model
pSrcImages	PVOID	Pointer to an array of source input images
nSrcArraySize	UINT	The size of the array referenced by pSrcImages. pSrcImages must reference an array with a size equal to the number of model inputs.
sInputNames	STRING	Input names separated by ";"
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image based on the output layer is created. To process the destination image, use e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage if necessary.
sOutputName	STRING	Specify the name of the layer for which the output is required. If empty, the output of one output layer is provided.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

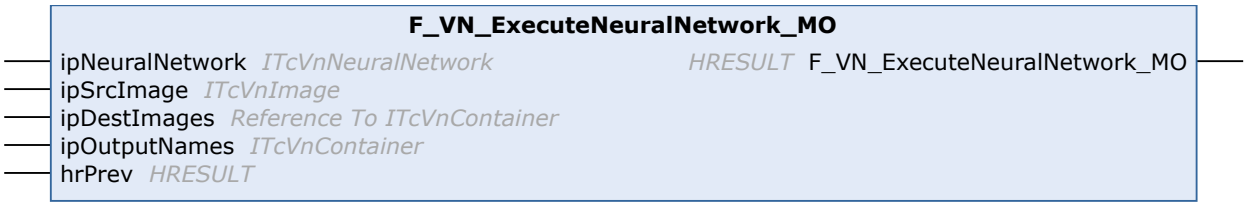
Required License

TC3 Neural Network Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.22.6 F_VN_ExecuteNeuralNetwork_MO



Execute a neural network using the provided image as input and provide the output of the specified output layers. The function doesn't perform any pre-processing or transformation.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_ExecuteNeuralNetwork_MO : HRESULT
VAR_INPUT
    ipNeuralNetwork : ITcVnNeuralNetwork;
    ipSrcImage      : ITcVnImage;
    ipDestImages    : Reference To ITcVnContainer;
    ipOutputNames   : ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipNeuralNetwork	ITcVnNeuralNetwork [▶ 445]	Neural network model
ipSrcImage	ITcVnImage [▶ 426]	Source image (The image layout must be adapted to match the model's expected input format using e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage functions if necessary.)
ipDestImages	Reference To ITcVnContainer [▶ 385]	Container of images containing the output from the specified layers.
ipOutputNames	ITcVnContainer [▶ 385]	Container of layer names whose outputs to be returned (ContainerType_Vector_String_SINT). If set to zero, the outputs of all unconnected output layers will be returned..
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

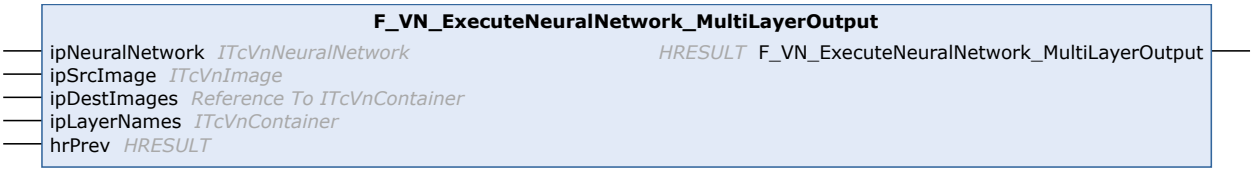
Required License

TC3 Neural Network Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.22.7 F_VN_ExecuteNeuralNetwork_MultiLayerOutput



This function is deprecated, please use F_VN_ExecuteNeuralNetwork_MO. Execute a neural network using the provided image as input and provide the output of the specified output layers. The function doesn't perform any pre-processing or transformation.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_ExecuteNeuralNetwork_MultiLayerOutput : HRESULT
VAR_INPUT
    ipNeuralNetwork : ITcVnNeuralNetwork;
    ipSrcImage      : ITcVnImage;
    ipDestImages    : Reference To ITcVnContainer;
    ipLayerNames    : ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipNeuralNetwork	ITcVnNeuralNetwork [▶ 445]	Neural network model
ipSrcImage	ITcVnImage [▶ 426]	Source image (The image layout must be adapted to match the model's expected input format using e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage functions if necessary.)
ipDestImages	Reference To ITcVnContainer [▶ 385]	Container of images containing the output from the specified layers.
ipLayerNames	ITcVnContainer [▶ 385]	Container of layer names whose outputs to be returned (ContainerType_Vector_String_SINT). If set to zero, the outputs of all unconnected output layers will be returned..
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

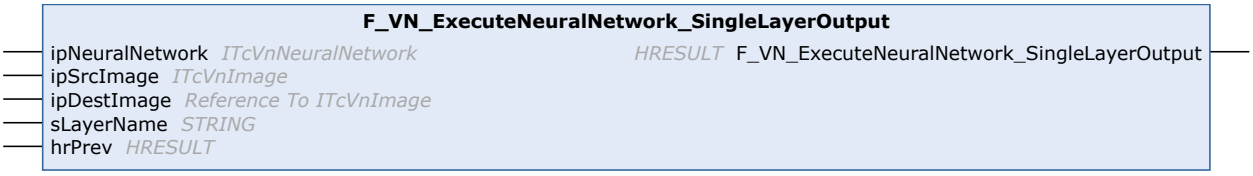
Required License

TC3 Neural Network Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.22.8 F_VN_ExecuteNeuralNetwork_SingleLayerOutput



This function is deprecated, please use F_VN_ExecuteNeuralNetwork_SO. Execute a neural network using the provided image as input and provide the output of the specified output layer. The function doesn't perform any pre-processing or transformation.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_ExecuteNeuralNetwork_SingleLayerOutput : HRESULT
VAR_INPUT
    ipNeuralNetwork : ITcVnNeuralNetwork;
    ipSrcImage      : ITcVnImage;
    ipDestImage     : Reference To ITcVnImage;
    sLayerName      : STRING;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipNeuralNetwork	ITcVnNeuralNetwork [▶ 445]	Neural network model
ipSrcImage	ITcVnImage [▶ 426]	Source image (The image layout must be adapted to match the model's expected input format using e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage functions if necessary.)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image based on the output layer is created. To process the destination image, use e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage if necessary.
sLayerName	STRING	Specify the name of the layer for which the output is required. If empty, the output of one output layer is provided.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

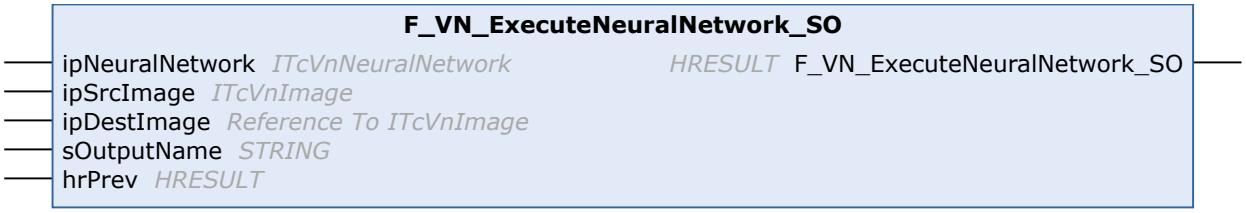
Required License

TC3 Neural Network Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.22.9 F_VN_ExecuteNeuralNetwork_SO



Execute a neural network using the provided image as input and provide the output of the specified output layer. The function doesn't perform any pre-processing or transformation.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_ExecuteNeuralNetwork_SO : HRESULT
VAR_INPUT
    ipNeuralNetwork : ITcVnNeuralNetwork;
    ipSrcImage      : ITcVnImage;
    ipDestImage     : Reference To ITcVnImage;
    sOutputName     : STRING;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipNeuralNetwork	ITcVnNeuralNetwork [▶ 445]	Neural network model
ipSrcImage	ITcVnImage [▶ 426]	Source image (The image layout must be adapted to match the model's expected input format using e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage functions if necessary.)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image based on the output layer is created. To process the destination image, use e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage if necessary.
sOutputName	STRING	Specify the name of the layer for which the output is required. If empty, the output of one output layer is provided.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

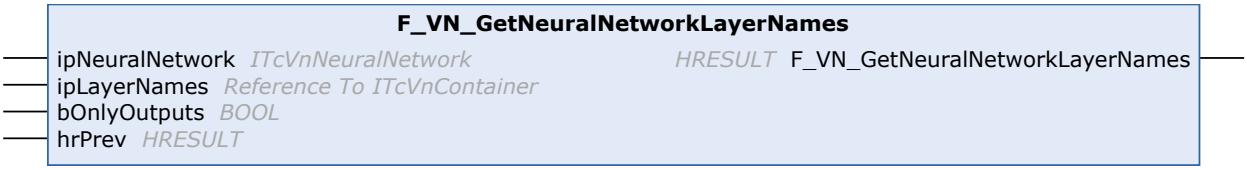
Required License

TC3 Neural Network Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.22.10 F_VN_GetNeuralNetworkLayerNames



Get the name of all layers in the neural network model. If the bOnlyOuput flag is set to true, only the unconnected output layer names are returned.

Syntax

Definition:

```
FUNCTION F_VN_GetNeuralNetworkLayerNames : HRESULT
VAR_INPUT
    ipNeuralNetwork : ITcVnNeuralNetwork;
    ipLayerNames    : Reference To ITcVnContainer;
    bOnlyOutputs    : BOOL;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipNeuralNetwork	ITcVnNeuralNetwork [▶ 445]	Neural network model
ipLayerNames	Reference To ITcVnContainer [▶ 385]	Container with the layers names (ContainerType_Vector_String_SINT)
bOnlyOutputs	BOOL	If true, only the unconnected output layer names are returned.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

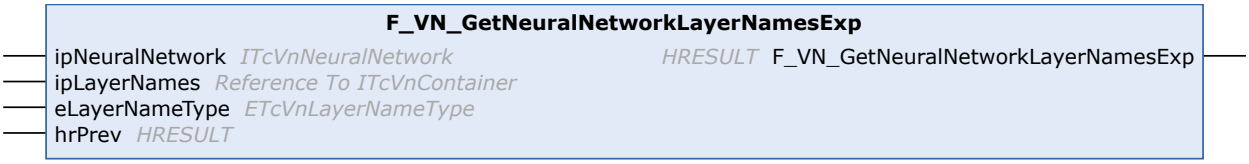
Required License

TC3 Neural Network Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.22.11 F_VN_GetNeuralNetworkLayerNamesExp



Get the name of model layers in the neural network model. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_GetNeuralNetworkLayerNamesExp : HRESULT
VAR_INPUT
    ipNeuralNetwork : ITcVnNeuralNetwork;
    ipLayerNames    : Reference To ITcVnContainer;
    eLayerNameType  : ETcVnLayerNameType;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipNeuralNetwork	ITcVnNeuralNetwork [▶ 445]	Neural network model
ipLayerNames	Reference To ITcVnContainer [▶ 385]	Container with the layers names (ContainerType_Vector_String_SINT)
eLayerNameType	ETcVnLayerNameType [▶ 238]	Layer name type
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

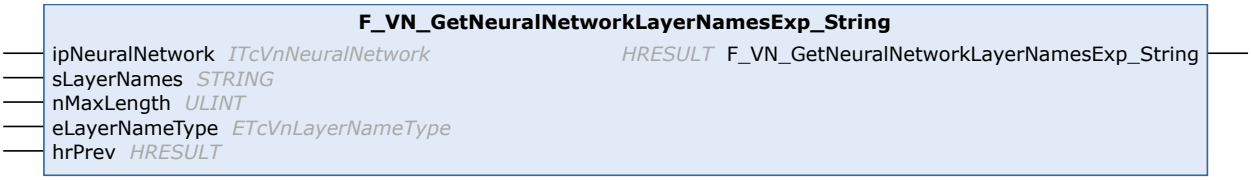
Required License

TC3 Neural Network Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.22.12 F_VN_GetNeuralNetworkLayerNamesExp_String



Get the name of model layers in the neural network model. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_GetNeuralNetworkLayerNamesExp_String : HRESULT
VAR_INPUT
    ipNeuralNetwork : ITcVnNeuralNetwork;
    sLayerNames     : STRING;
    nMaxLength      : ULINT;
    eLayerNameType  : ETcVnLayerNameType;
    hrPrev          : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipNeuralNetwork	ITcVnNeuralNetwork [▶ 445]	Neural network model
sLayerNames	STRING	Layer names separated by ";". Make sure to choose a sufficient STRING size!
nMaxLength	ULINT	Maximum string length to export (including 0 termination). If the container content is longer, the string is cut off at nMaxLength - 1 and 0 termination is appended. In this case, S_FALSE is returned.
eLayerNameType	ETcVnLayerNameType [▶ 238]	Layer name type
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

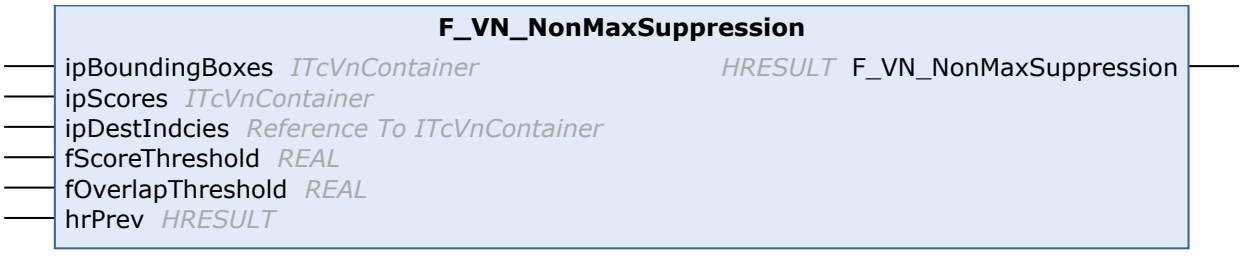
Required License

TC3 Neural Network Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.22.13 F_VN_NonMaxSuppression



Perform non-maximum suppression on given bounding boxes and corresponding scores.

Syntax

Definition:

```
FUNCTION F_VN_NonMaxSuppression : HRESULT
VAR_INPUT
    ipBoundingBoxes   : ITcVnContainer;
    ipScores          : ITcVnContainer;
    ipDestIndcies     : Reference To ITcVnContainer;
    fScoreThreshold   : REAL;
```

```
    fOverlapThreshold : REAL;
    hrPrev            : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipBoundingBoxes	ITcVnContainer [▶ 385]	Container of bounding boxes to apply the non maximum supresion (ContainerType_Vector_TcVnRectangle_DINT)
ipScores	ITcVnContainer [▶ 385]	Container of coreeponding score for each bounding box (ContainerType_Vector_REAL)
ipDestIndcies	Reference To ITcVnContainer [▶ 385]	Container of the kept indcies after apply the non maximum suppression (ContainerType_Vector_DINT)
fScoreThreshold	REAL	A threshold [0.0 to 1.0] used to filter the bounding boxes by score
fOverlapThreshold	REAL	A threshold [0.0 to 1.0] that determines the minimum degree of intersection over union (IoU) required for bounding boxes to be considered overlapping
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

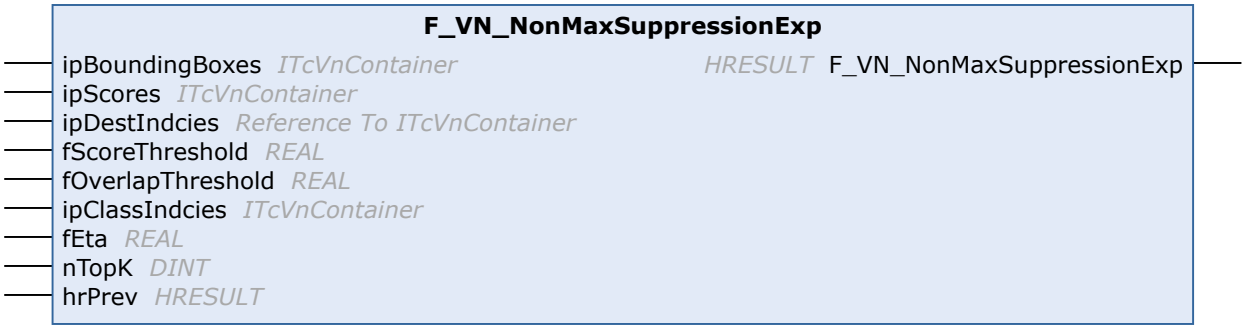
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.22.14 F_VN_NonMaxSuppressionExp



Perform non-maximum suppression on given bounding boxes and corresponding scores. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_NonMaxSuppressionExp : HRESULT
VAR_INPUT
    ipBoundingBoxes   : ITcVnContainer;
    ipScores          : ITcVnContainer;
    ipDestIndcies     : Reference To ITcVnContainer;
    fScoreThreshold   : REAL;
    fOverlapThreshold : REAL;
```

```
    ipClassIndcies    : ITcVnContainer;
    fEta              : REAL;
    nTopK             : DINT;
    hrPrev            : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipBoundingBoxes	ITcVnContainer [▶ 385]	Container of bounding boxes to apply the non maximum supresion (ContainerType_Vector_TcVnRectangle_DINT)
ipScores	ITcVnContainer [▶ 385]	Container of coreeponding score for each bounding box (ContainerType_Vector_REAL)
ipDestIndcies	Reference To ITcVnContainer [▶ 385]	Container of the kept indcies after apply the non maximum suppression (ContainerType_Vector_DINT)
fScoreThreshold	REAL	A threshold [0.0 to 1.0] used to filter the bounding boxes by score
fOverlapThreshold	REAL	A threshold [0.0 to 1.0] that determines the minimum degree of intersection over union (IoU) required for bounding boxes to be considered overlapping
ipClassIndcies	ITcVnContainer [▶ 385]	Container of corresponding set of class indcies (ContainerType_Vector_DINT)
fEta	REAL	An adjustable threshold [0.0 to 1.0] used to dynamically reduce the overlap threshold during processing. Setting it to 1 maintains a constant value for the overlap threshold
nTopK	DINT	The maximum number of bounding boxes with the highest scores to undergo the suppression algorithm (must be >= 0). Setting it to zero applies the algorithm to all bounding boxes
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

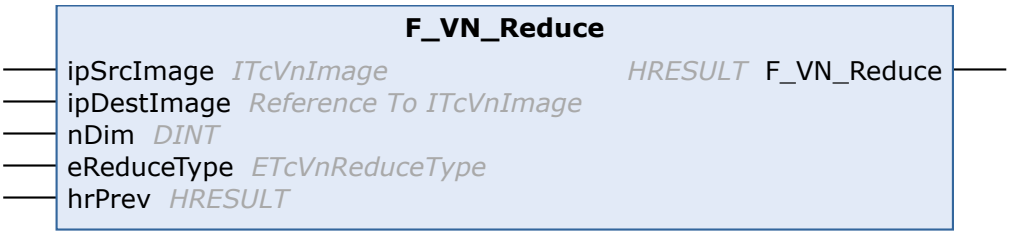
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.22.15 F_VN_Reduce



Reduce ipSrcImage by applying the specified operation across rows/columns to obtain a single row/column/channel with the resulting values.

Syntax

Definition:

```
FUNCTION F_VN_Reduce : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    nDim        : DINT;
    eReduceType : ETcVnReduceType;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT or REAL elements, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image. If ipSrcImage is of type USINT and the reduction type is SUM or AVG, ipDestImage will be of type REAL; otherwise, it retains the same type as ipSrcImage (An appropriate single channel destination image will be created if required.)
nDim	DINT	Index along which the ITcVnImage is reduced: 0 reduces the matrix to a single row, 1 reduces it to a single column, and 2 reduces to a single channel
eReduceType	ETcVnReduceType [▶ 236]	Reduction operation
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.22.16 F_VN_ReduceArg

Finds indices of max/min elements along provided axis and reduce the ipSrcImage to a single row/column/channel.

Syntax

Definition:

```
FUNCTION F_VN_ReduceArg : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    nDim        : DINT;
    eReduceType : DINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image of type DINT (An appropriate single channel destination image will be created if required.)
nDim	DINT	Index along which the ipSrcImage is reduced: 0 reduces the matrix to a single row, 1 reduces it to a single column, and 2 reduces to a single channel
eReduceType	DINT	Reduction operation REDUCE_MAX or REDUCE_MIN
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.22.17 F_VN_ReshapeImage

Changes the shape of an ITcVnImage. The total number of elements in the new shape must be equal to the total number of elements in ipSrcImage.

Syntax

Definition:

```
FUNCTION F_VN_ReshapeImage : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    aNewShape   : TcVnArray33_UDINT;
END_VAR
VAR_INPUT
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (1 channel)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate single channel destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

In/Outputs

Name	Type	Description
aNewShape	TcVnArray33_UDINT [▶ 173]	Specifies the shape of ipDestImage. First entry is the number of dimensions (max. 32), followed by the sizes of each dimension. If one dimension is set to zero, its value will be automatically calculated based on the total elements of ipSrcImage (Only one dimension can be set to zero.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.22.18 F_VN_Sigmoid

Apply the standard logistic function 1 / (1 + exp(-x)) element-wise to a single- or multi-channel REAL image.

Syntax

Definition:

```
FUNCTION F_VN_Sigmoid : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (REAL)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.22.19 F_VN_SigmoidExp

Apply the logistic function 1 / (1 + exp(-fSteepness * (x - fShift))) element-wise to a single- or multi-channel REAL image. (expert function)

Syntax

Definition:

```
FUNCTION F_VN_SigmoidExp : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    fShift      : REAL;
    fSteepness  : REAL;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (REAL)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate destination image will be created if required.)
fShift	REAL	Determines the intensity level around which the logistic function transitions.
fSteepness	REAL	Controls the sharpness of the transition.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.22.20 F_VN_SoftMax

Apply softmax function pixel-wise to a single channel REAL image.

Syntax

Definition:

```
FUNCTION F_VN_SoftMax : HRESULT
VAR_INPUT
    ipSrcImage  : ITcVnImage;
    ipDestImage : Reference To ITcVnImage;
    bFlatten    : BOOL;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (REAL, 1 channel)
ipDestImage	Reference To ITcVnImage [▶ 426]	Destination image (An appropriate single channel destination image will be created if required.)
bFlatten	BOOL	If true, the resulting image will be reshaped into a single row.
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.23 OCR

Diese Gruppe enthält Funktionen zur optischen Zeichenerkennung (**O**ptical **C**haracter **R**ecognition).

Die OCR-Funktionen [▶ 1612] identifizieren Zeichen in einem Bild und geben die erkannte Zeichenkette als String zurück. Die Klassifizierung basiert auf klassischen Machine Learning Modellen. Die Modelle werden trainiert bereitgestellt, so dass die Funktionen direkt angewendet werden können. Zusätzliche Einstellungen oder ein eigenes Training sind nicht erforderlich. Die unterstützen Zeichen und Fonts hängen von den im Training verwendeten Daten ab. Daher gibt es verschiedene Modelle deren Funktionsumfang, sowie die generellen Vorbedingungen und Anforderungen, im Folgenden beschrieben werden.

Für alle Zeichen, die in einem Bild erkannt werden, ordnen die Modelle die einzelnen Zeichen einer bekannten Klasse zu, bei der die Übereinstimmung am höchsten ist. Das Ergebnis besteht also immer aus den bekannten Zeichen des verwendeten Modelles. Eine Zurückweisung oder Aussortierung unbekannter Zeichen ist nicht enthalten.

Die Funktionen unterstützen nur einzeilige Zeichenfolgen, mehrere Zeilen müssen in einzelne ROIs unterteilt und mit mehreren Funktionsaufrufen gelesen werden.

Generelle Anforderungen zur Erkennung von Zeichen:

- Zeichenhöhe min. 20 Pixel
- Strichstärke min. 3 Pixel
- Punkte min. 3 x 3 Pixel
- Striche min. 3 x 6 Pixel
- Zeichenabstand / -trennung min. 4 Pixel
- Zeichen dürfen sich nicht überlappen
- Nur horizontale Ausrichtung / Anordnung der Zeichen max. ± 6° Abweichung
- Der Strich / Kontur eines Zeichens darf nicht unterbrochen sein
- Die Zeichen dürfen nicht gespiegelt oder auf den Kopf gestellt sein

Generelle Anforderungen an das Bild:

- ROI nur mit den zu erkennenden Zeichen und einer störungsfreien Zone um die Zeichen
- Innerhalb der ROI dürfen die Zeichen nicht von einer weiteren Kontur wie z.B. einem Rechteck umschlossen sein
- Guter Kontrast zwischen Zeichen und Hintergrund
- Homogener, nicht verrauschter oder gestörter, intransparenter Hintergrund

Anforderungen an die Fonts:

- Nur proportionale Schriftarten mit gleichen Zeichenbreiten
- Größere Lücken werden immer nur als ein Leerzeichen erkannt
- Nur serifenlose Schriftarten wie z.B. Arial, Tahoma, Courier, Univers, Frutiger, Verdana, OCR-B
- Keine gemischten Schriftarten

- Keine Punktdruck (dot print) oder kursiv (italic) Fonts

Modelle:

Das Enum ETcVnOcrModelType [▶ 222] ermöglicht den Zugriff auf die folgenden Modelle:

- `TCVN_OMT_NUMBERS`
  - ermöglich die Klassifizierung von Zahlen
  - enthält die Zeichen von `0-9`
- `TCVN_OMT_NUMBERS_SC`
  - ermöglich die Klassifizierung von Zahlen und Sonderzeichen
  - enthält die Zeichen von `0-9` und sechs Sonderzeichen `. / - : = +`
- `TCVN_OMT_UCLETTERS`
  - ermöglich die Klassifizierung von Großbuchstaben
  - enthält die Zeichen von `A-Z`
- `TCVN_OMT_NUMBERS_SC_UCLETTERS`
  - ermöglich die Klassifizierung von Zahlen, Sonderzeichen und Großbuchstaben
  - enthält die Zeichen von `0-9`, die sechs Sonderzeichen `. / - : =` + und `A-Z`

Bei der Verwendung des kombinierten Modells (`TCVN_OMT_NUMBERS_SC_UCLETTERS`) kann es, aufgrund der großen Ähnlichkeit von bestimmten Zeichen, zu Verwechselungen kommen. Beispiele dafür sind O und 0, S und 5 sowie B und 8.

Wenn bekannt ist an welcher Stelle eine Zahl oder ein Buchstabe steht, sollte alternativ die Expert-Funktion mit der Formatierungsvorgabe (`sPattern`) und der Kombination von den separaten Modellen verwendet werden. Alternativ können im Anschluss die Zeichen des Ergebnisstrings (`ipCharacters`) auch einzeln analysiert werden, so dass man individuell entscheiden kann ob z.B. anstatt O auch eine 0 akzeptiert wird.

Funktionen initialisieren

Um die OCR-Funktion mit einem oder mehreren Modellen verwenden zu können, muss die Funktion zuvor über den Baustein FB_VN_InitializeFunction [▶ 1640] mit den jeweiligen Modellen initialisiert werden.

Die Dauer für das Laden der Modelle ist abhängig von der Leistungsfähigkeit des verwendeten IPC, sowie der Modellgröße und kann mehrere hundert Millisekunden, bis hin zu mehreren Sekunden dauern. Daher ist bei kürzeren Taskzykluszeiten mit Zykluszeitüberschreitungen zu rechnen.

Wenn das kombinierte Modell `TCVN_OMT_NUMBERS_SC_UCLETTERS` verwendet werden soll, muss aufgrund der Dateigröße der Router Speicher auf mindestens 512 MB gesetzt werden.

Die genaue Größe ist abhängig von der sonstigen Verwendung des Router Speichers und evtl. vorhandener Fragmentierung. Wenn der Baustein `FB_VN_InitializeFunction` z.B. einen Return Code `0x80004005` zurückgibt, deutet das auf zu wenig Router Speicher hin. Daher sollte zuerst der Router Speicher geprüft und ggf. vergrößert werden. Nach dem Laden eines Modells wird ein Großteil des Speichers wieder freigegeben und steht für die Anwendung zur Verfügung.

Interpretation des HRESULT

- Bei der Standardfunktion wird `S_OK` zurückgegeben, wenn Zeichen auf dem Bild erkannt wurden. Bei der Exp-Funktion hängt es davon ab, ob ein `sPattern` angegeben wurde. Wenn keine Formatierungsvorgabe angegeben wurde, entspricht die Rückgabe der Standardfunktion. Wenn eine Formatierungsvorgabe übergeben wurde, wird nur `S_OK` zurückgegeben, wenn die erkannten Zeichen mit der Pattern-Vorgabe übereinstimmen.
- Wenn die Standardfunktion erfolgreich ausgeführt wurde und keine Zeichen gefunden wurden oder bei der Exp-Funktion die Zeichen nicht mit der Pattern-Vorgabe übereinstimmen, wird `S_FALSE` zurückgegeben.
- Wenn bei der Expert-Funktion eine `sPattern` Angabe gemacht wird, zu der das übergebene Modell fehlt, wird ein `HRESULT = ADSERR_DEVICE_NOTINIT` zurückgegeben.

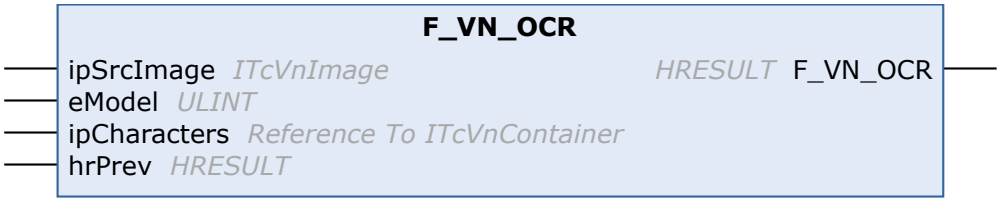
- Wenn der Standardfunktion mehrere Modelle übergeben werden, wird immer ein `HRESULT = ADSERR_DEVICE_INVALIDPARM` zurückgegeben, da die Funktion pro Aufruf nur ein Modell unterstützt. Bei der Expert-Funktion hängt es davon ab, ob ein `sPattern` angegeben wurde. Wenn `sPattern` leer ist, wird immer nur das `TCVN_OMT_NUMBERS_SC_UCLETTERS` Model verwendet. Wenn dieses Modell nicht übergeben wurde, wird ein `HRESULT = ADSERR_DEVICE_INVALIDPARM` zurückgegeben.

Beispiele für die Verwendung von sPattern

Die F_VN_OCRExp [▶ 1614] stellt mit den Parametern `sPattern` und `eOcrOptions` zusätzliche Möglichkeiten bereit. Je nach Kombination der Zeichen auf dem Eingangsbild, den Angaben von `sPattern` und `eOcrOptions` und den erkannten Zeichen, wird bei einer Nicht-Übereinstimmung `S_FALSE` zurückgegeben. Danach können die Zeichen des Ergebnis-Strings (`ipCharacters`) auch einzeln analysiert werden, da z.B. bei abweichender Länge auch `S_FALSE` zurückgegeben wird. Die Beispiele beziehen sich auf die Verwendung des `TCVN_OMT_NUMBERS_SC_UCLETTERS` Modells.

Zeichen auf dem ipSrcImage	sPattern	eOcrOptions	ipCharacters	HRESULT
12/34	dd.dd		12/34	S_OK
12534	dd.dd		12534	S_OK
12/34	dd!dd		1234	S_OK
12 34	dd!d		124	S_OK
12534	dd!dd		1234	S_OK
12 34	dd_dd		12 34	S_OK
AB12/	uudd#		AB12/	S_OK
12 34	dddd	WITHBLANKS	12 34	S_OK
12/4	dd#d	WITHBLANKS	12/4	S_OK
12 34	dd_dd	WITHBLANKS	12 34	S_OK
12 34 56	dd_dddd	WITHBLANKS	12 34 56	S_OK
12 3 4	dd.dd	WITHBLANKS	12 3 4	S_FALSE
12 34	dd!dd		124	S_FALSE
12 34	dd.dd		1234	S_FALSE
1234	!dddd		234	S_FALSE
12 34	dd_dd		12.34	S_FALSE
12 3	dd_dd		123	S_FALSE
12 3 4	dd_d_d		123 4	S_FALSE
AB12/	uddd#		AB12/	S_FALSE
AB12/	uuddd		AB12/	S_FALSE

6.1.4.23.1 F_VN_OCR



Detects and recognizes characters in a binary image (white characters on black background).
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_OCR : HRESULT
VAR_INPUT
    ipSrcImage    : ITcVnImage;
```

```
    eModel       : ULINT;
    ipCharacters : Reference To ITcVnContainer;
    hrPrev       : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, 1 channel, binary)
eModel	ULINT	Selection of OCR model types (ETcVnOcrModelType)
ipCharacters	Reference To ITcVnContainer [▶ 385]	Returns the recognized characters (ContainerType_Vector_String_SINT)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_OCR` detektiert und erkennt Zeichen in einem Binärbild und gibt die erkannten Zeichen in einem Container zurück.

Parameter

Eingangsbild

Das Eingangsbild was an `ipSrcImage` übergeben wird, muss ein 1-kanaliges Binärbild sein. Es sind nur weiße Zeichen auf schwarzem Hintergrund zulässig.

Modell

Ein Modell-Typ von ETcVnOcrModelType [▶ 222], welches bei einem Funktionsaufruf zur Klassifizierung der Zeichen verwendet werden soll, muss an `eModel` übergeben werden. Diese Funktion unterstützt nur die Verwendung von einem Modell pro Funktionsaufruf.

Um ein Modell verwenden zu können, muss dieses zuvor mit dem Baustein FB_VN_InitializeFunction [▶ 1640] initialisiert werden.

Erkannte Zeichen

Die erkannten Zeichen werden über die Referenz `ipCharacters` zurückgegeben.

Expert-Parameter

Die Expert-Variante F_VN_OCRExp [▶ 1615] enthält zusätzliche Parameter.

Anwendung

Ein Aufruf der OCR-Funktion, mit einem Modell zur Erkennung von Zahlen, mit anschließendem String-Export sieht z.B. so aus:

```
hr := F_VN_OCR(
    ipSrcImage   := ipBinaryImage,
    eModel       := TCVN_OMT_NUMBERS,
    ipCharacters := ipCharactersResults,
    hrPrev       := hr);

// Export character to string
hr := F_VN_ExportSubContainer_String(ipCharactersResults, 0, sText, 255, hr);
```

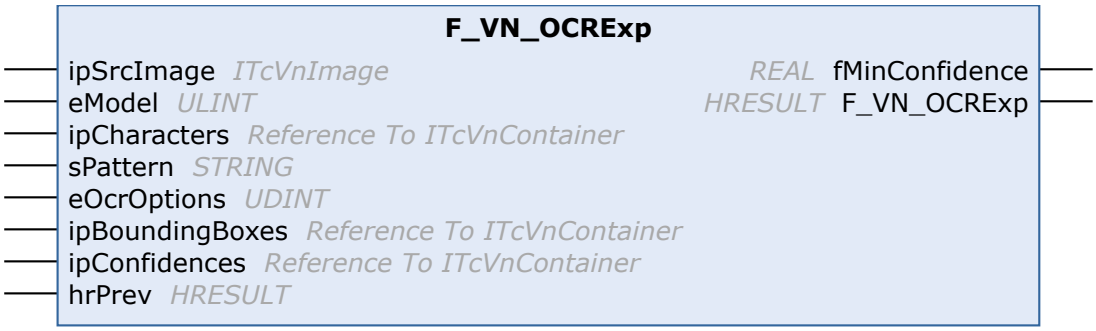
Required License

TC3 Vision OCR

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.23.2 F_VN_OCRExp



Detects and recognizes characters in a binary image (white characters on black background). (expert function)
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
FUNCTION F_VN_OCRExp : HRESULT
VAR_INPUT
    ipSrcImage      : ITcVnImage;
    eModel          : ULINT;
    ipCharacters    : Reference To ITcVnContainer;
    sPattern        : STRING;
    eOcrOptions     : UDINT;
    ipBoundingBoxes : Reference To ITcVnContainer;
    ipConfidences   : Reference To ITcVnContainer;
    hrPrev          : HRESULT;
END_VAR
VAR_OUTPUT
    fMinConfidence  : REAL;
END_VAR
```

Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT, 1 channel, binary)
eModel	ULINT	Selection of OCR model types (ETcVnOcrModelType)
ipCharacters	Reference To ITcVnContainer [▶ 385]	Returns the recognized characters (ContainerType_Vector_String_SINT)
sPattern	STRING	String pattern containing the format in which characters are presented
eOcrOptions	UDINT	Specifies which options must be applied to the function (ETcVnOcrOptions)
ipBoundingBoxes	Reference To ITcVnContainer [▶ 385]	Returns the corresponding bounding boxes of the recognized characters (ContainerType_Vector_TcVnRectangle_DINT, optional, set to 0 if not required)
ipConfidences	Reference To ITcVnContainer [▶ 385]	Returns the corresponding classification confidences of the recognized characters (ContainerType_Vector_REAL, optional, set to 0 if not required)
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Outputs

Name	Type	Description
fMinConfidence	REAL	Returns the minimum value of the confidences

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_OCRExp` ist die Expert-Variante von F_VN_OCR [▶ 1612]. Sie enthält zusätzliche Parameter.

Parameter

Eingangsbild

Das Eingangsbild was an `ipSrcImage` übergeben wird, muss ein 1-kanaliges Binärbild sein. Es sind nur weiße Zeichen auf schwarzem Hintergrund zulässig.

Modell

Die Modell-Typen von ETcVnOcrModelType [▶ 222], die bei einem Funktionsaufruf zur Klassifizierung der Zeichen verwendet werden sollen, müssen an `eModel` übergeben werden. Um mehrere Modelle mit einem Funktionsaufruf verwenden zu können, muss ein Zeichenkettenmuster über `sPattern` angegeben werden. Die Modelle können mit einem `OR` verkettet werden.

Um die einzelnen Modelle verwenden zu können, müssen diese zuvor mit dem Baustein FB_VN_InitializeFunction [▶ 1640] initialisiert werden.

Erkannte Zeichen

Die erkannten Zeichen werden über die Referenz `ipCharacters` zurückgegeben.

Zeichenkettenmuster

Über `sPattern` kann ein Zeichenkettenmuster, mit dem erwarteten Format der zu erkennenden Zeichenkette, übergeben werden.

Unterstützte Formatierungen und deren Bedeutung:

- `!` (Ausrufezeichen) zum Ignorieren von Zeichen (gilt nicht für Leerzeichen)
- `.` (Punkt) für ein beliebiges Zeichen (gilt nicht für Leerzeichen)
  Die Verwendung des Punkts wird nur mit dem Modell TCVN_OMT_NUMBERS_SC_UCLETTERS unterstützt, ansonsten wirkt sich der Punkt wie ein Ausrufezeichen aus und es wird ein S_FALSE ausgegeben.
- `_` (Unterstrich) für ein Leerzeichen (unabhängig von `eOcrOptions`)
- `d` (Zahl) für eine erwartete Zahl von `0-9`
- `#` (Sonderzeichen) für ein erwartetes Sonderzeichen, unterstützt werden `. / - : = +`
- `u` (Großbuchstabe) für einen erwarteten Großbuchstaben von `A-Z`

Optionen

Mit `eOcrOptions` können Optionen (ETcVnOcrOptions [▶ 223]) angegeben werden, die die Funktionsweise und Ergebnisausgabe beeinflussen. Wenn keine Optionen benötigt werden, muss eine 0 übergeben werden.

Bounding Boxes

Über die Referenz `ipBoundingBoxes` können optional die entsprechenden Bounding Boxes der erkannten Zeichen zurückgegeben werden. Wenn keine Bounding Boxes benötigt werden, kann der Wert 0 übergeben werden.

Konfidenzen

Über die Referenz `ipConfidences` können optional die entsprechenden Klassifizierungskonfidenzen der erkannten Zeichen zurückgegeben werden. Wenn keine Konfidenzen benötigt werden, kann der Wert 0 übergeben werden.

Minimale Konfidenz

Der optionale Rückgabewert `fMinConfidence` gibt den minimalen Konfidenz-Wert aller gefundenen Zeichen zurück.

Anwendung

Ein Aufruf der OCRExp-Funktion, mit zwei Modellen zur Erkennung von Zahlen und Großbuchstaben, mit anschließendem String-Export sieht z.B. so aus:

```
sPattern := 'dd!uuu!dddd'; // e.g. 02-FEB-2024

hr := F_VN_OCRExp(
    ipSrcImage      := ipBinaryImage,
    eModel          := TCVN_OMT_NUMBERS OR TCVN_OMT_UCLETTERS,
    ipCharacters    := ipCharactersResults,
    sPattern        := sPattern,
    eOcrOptions     := eOcrOptions,
    ipBoundingBoxes := ipBoundingBoxes,
    ipConfidences   := ipConfidences,
    hrPrev          := hr,
    fMinConfidence  => fMinConfidence);

// Export character to string
hr := F_VN_ExportSubContainer_String(ipCharactersResults, 0, sText, 255, hr);

// Get the bounding box of the first element
hr := F_VN_GetAt_TcVnRectangle_DINT(ipBoundingBoxes, stRectangle, 0, hr);

// Get the Confidence value of the first element
hr := F_VN_GetAt_REAL(ipConfidences, fConfidence, 0, hr);
```

Zur Auswertung der Konfidenz-Werte oder zum Zeichnen von Bounding Boxen kann über `GetAt` auf einzelne Elemente der Container zugegriffen werden. Wenn man jedoch auf mehrere oder alle Elemente eines Containers zugreifen möchte, empfiehlt es sich, eine Schleife zu programmieren oder den `ipConfidences` Container komplett als Array zu exportieren.

Required License

TC3 Vision OCR

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.24 Texture Analysis

Diese Gruppe enthält Funktionen zur Texturanalyse eines Bildes.

Funktionen

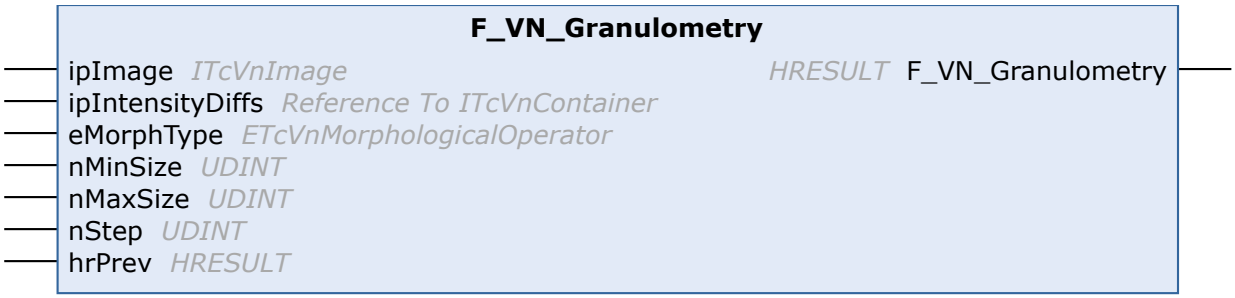
Granulometrische Eigenschaften

- F_VN_Granulometry [▶ 1617]

Haralick Merkmale

- F_VN_HaralickFeatures [▶ 1619]

6.1.4.24.1 F_VN_Granulometry



Computes the granulometry of a textured image using morphological operations.

Syntax

Definition:

```
FUNCTION F_VN_Granulometry : HRESULT
VAR_INPUT
    ipImage          : ITcVnImage;
    ipIntensityDiffs : Reference To ITcVnContainer;
    eMorphType       : ETcVnMorphologicalOperator;
    nMinSize         : UDINT;
    nMaxSize         : UDINT;
    nStep            : UDINT;
    hrPrev           : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Source image (1 channel)
ipIntensityDiffs	Reference To ITcVnContainer [▶ 385]	Returns the intensity differences between successive structuring element sizes (CTcVnContainer_Vector_LREAL), which can be used as features for texture description.
eMorphType	ETcVnMorphologicalOperator [▶ 220]	Only opening or closing are supported.
nMinSize	UDINT	Min structuring element size (odd, >= 3)
nMaxSize	UDINT	Max structuring element size (>= nMinSize)
nStep	UDINT	Step, by which nMinSize is incremented until nMaxSize is exceeded (even, >= 2).
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_Granulometry` berechnet die granulometrischen Eigenschaften eines Bildes. Diese Werte geben einen Hinweis auf die Größenverteilung verschiedener Bildbereiche.

Parameter

Bild

Das Eingangsbild `ipImage`.

Merkmale

Die granulometrischen Merkmale werden als Container über die Referenz `ipIntensityDiffs` zurückgegeben.

Operator-Typ

Der Typ der morphologischen Operatoren wird über `eMorphType` als `TCVN_MO_OPENING` oder als `TCVN_MO_CLOSING` festgelegt.

Operator-Größe

Die Größe der morphologischen Operatoren wird über das Minimum `nMinSize`, das Maximum `nMaxSize` und die Schrittgröße `nStep` festgelegt. Die Operatoren müssen eine ungerade Größe haben und die Schrittweite muss gerade sein.

Anwendung

Die Berechnung von granulometrischen Merkmalen mit Closing-Operatoren in den Größen von drei, sieben und elf sieht z.B. so aus:

```
hr := F_VN_Granulometry(
    ipImage             :=  ipImage,
    ipIntensityDiffs    :=  ipIntensityDiffs,
    eMorphType          :=  TCVN_MO_CLOSING,
    nMinSize            :=  3,
    nMaxSize            :=  11,
    nStep               :=  4,
    hrPrev              :=  hr);
```

Verwandte Funktionen

- F_VN_HaralickFeatures [▶ 1619]

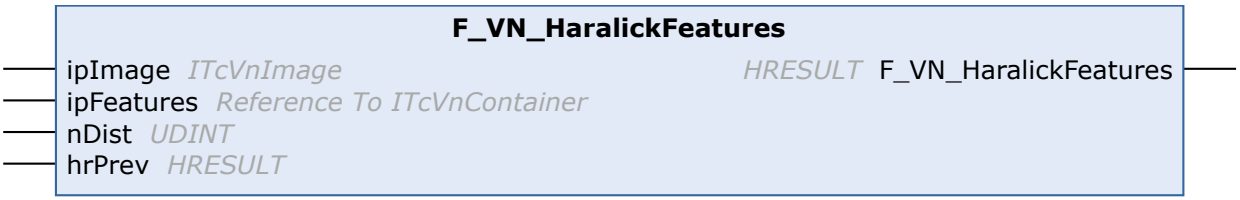
Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.4.24.2 F_VN_HaralickFeatures



Computes the Haralick features of a graylevel image, which describe the texture.

Syntax

Definition:

```
FUNCTION F_VN_HaralickFeatures : HRESULT
VAR_INPUT
    ipImage     : ITcVnImage;
    ipFeatures : Reference To ITcVnContainer;
    nDist       : UDINT;
    hrPrev      : HRESULT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Source image (1 channel, USINT)
ipFeatures	Reference To ITcVnContainer [▶ 385]	Returns the computed Haralick features (CTcVnContainer_Vector_LREAL). The returned feature set contains the mean and standard deviation for each of the first 13 Haralick features, computed for 0, 45, 90, 135 degrees. So there will be 26 features in total (f1 mean, f1 stdDev, f2 mean, f2 stdDev, ..., f13 mean, f13 stdDev).
nDist	UDINT	Distance
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Funktion `F_VN_HaralickFeatures` berechnet die Haralick-Eigenschaften eines Bildes. Diese Werte geben einen Hinweis auf die Struktur des Bildes.

Parameter

Bild

Das Eingangsbild `ipImage`.

Merkmale

Die Haralick-Merkmale werden als Container über die Referenz `ipFeatures` zurückgegeben.

Distanz

Die Distanz zwischen benachbarten Pixeln in der Gray-Level Co-occurrence Matrix (GLCM) wird als `nDist` festgelegt.

Anwendung

Die Berechnung von Haralick-Merkmalen mit einer Distanz von 5 sieht z.B. so aus:

```
hr := F_VN_HaralickFeatures(
    ipImage     :=  ipImage,
    ipFeatures  :=  ipFeatures,
    nDist       :=  5,
    hrPrev      :=  hr);
```

Verwandte Funktionen

- F_VN_Granulometry [▶ 1617]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5 Function Blocks

Die Kommunikation mit GigE Vision Kameras oder mit dem Dateisystem sowie komplexeren Bildverarbeitungsfunktionen wird durch Funktionsblöcke abgebildet. Funktionsblöcke werden durch das Präfix „FB_VN_“ gekennzeichnet.

Gruppierung von Funktionsblöcken

Alle Funktionsblöcke der TwinCAT Vision Bibliothek sind thematisch in folgenden Gruppen sortiert:

- Camera Register Access [▶ 1621]
- File Access [▶ 1638]
- Image Acquisition [▶ 1662]
- Image Processing [▶ 1703]

6.1.5.1 License Overview

Folgend ist die Zugehörigkeit der TwinCAT Vision Funktionsblöcke zu den verfügbaren Lizenzen (siehe Lizenzmodell [▶ 12]) aufgelistet. Daraus ist ersichtlich, welche Lizenz für die Anwendung bestimmter Funktionsblöcke benötigt wird.

Die Basis-Lizenz TC3 Vision Base (TF7100) wird immer benötigt.

TC3 Vision Base

FB_VN_FileSourceControl [▶ 1664]

FB_VN_GevCameraControl [▶ 1672]

FB_VN_InitializeFunction [▶ 1640]

FB_VN_ReadCalibrationPattern [▶ 1642]

FB_VN_ReadCalibrationResult [▶ 1644]

FB_VN_ReadContainer [▶ 1646]

FB_VN_ReadImage [▶ 1648]

FB_VN_ReadMemory [▶ 1623]

FB_VN_ReadMlModel [▶ 1650]

FB_VN_ReadNeuralNetwork [▶ 1660]

FB_VN_ReadRegister_REAL [▶ 1624]

FB_VN_ReadRegister_UDINT [▶ 1626]

FB_VN_ReadRegister_ULINT [▶ 1628]

FB_VN_SimpleCameraControl [▶ 1697]

FB_VN_WriteCalibrationResult [▶ 1652]

FB_VN_WriteContainer [▶ 1654]

FB_VN_WriteImage [▶ 1656]

FB_VN_WriteMemory [▶ 1630]

FB_VN_WriteMlModel [▶ 1658]

FB_VN_WriteRegister_REAL [▶ 1632]

FB_VN_WriteRegister_UDINT [▶ 1634]

FB_VN_WriteRegister_ULINT [▶ 1636]

TC3 Vision Matching

FB_VN_GeneralizedHoughBallard [▶ 1703]

FB_VN_SSIM [▶ 1708]

6.1.5.2 Camera Register Access

Diese Gruppe enthält Funktionsblöcke zum Lesen und Schreiben von Kamera-Registern oder Speicherbereichen.

Funktionsblöcke

UDINT-Register (4 Byte)

- FB_VN_ReadRegister_UDINT [▶ 1626]
- FB_VN_WriteRegister_UDINT [▶ 1634]

ULINT-Register (8 Byte)

- FB_VN_ReadRegister_ULINT [▶ 1628]
- FB_VN_WriteRegister_ULINT [▶ 1636]

REAL-Register (4 Byte)

- FB_VN_ReadRegister_REAL [▶ 1624]
- FB_VN_WriteRegister_REAL [▶ 1632]

Größere Speicherbereiche

- FB_VN_ReadMemory [▶ 1623]
- FB_VN_WriteMemory [▶ 1630]

Parameter

Die folgenden Parameter funktionieren bei allen Funktionsblöcken dieser Gruppe gleich:

Adresse

Die Adresse `nAddress` gibt das zu lesende oder zu speichernde Register der Kamera an. Der Typ GVCP_REGISTER_ADDRESS ist lediglich ein Alias auf UDINT; sodass die Adresse 4 Byte (bzw. 32 Bit) lang sein muss. Bei größeren Speicherbereichen der Kamera (siehe FB_VN_ReadMemory [▶ 1623]) zeigt die Adresse auf den Anfang des zu lesenden oder zu speichernden Bereichs.

Funktion ausführen

Die Trigger für Schreib- und Lese-Befehle sind `bWrite` und `bRead`. Sie werden durch eine steigende Flanke (Wechsel von `FALSE` nach `TRUE`) ausgelöst. Dabei werden die relevanten Eingangsparameter intern gespeichert, sodass eine Änderung der Parameter während der Ausführung keinen Einfluss hat. Ebenso hat eine fallende Flanke der Trigger keine Auswirkung.

> **Funktionsblöcke aufrufen**
>
> Beachten Sie, dass das Beschreiben von Parametern bei Funktionsblöcken nur eine Auswirkung hat, wenn dieser auch aufgerufen wird.

Timeout

Um in unbekannten Fehlerfällen nicht ewig auf die Ausführung des Funktionsblocks zu warten, kann mit `nTimeout` die maximal erlaubte Ausführungszeit festgelegt werden. Wenn diese Zeit ohne erfolgreiches Ergebnis nach Auslösen des Befehlstriggers überschritten wird, wird die Ausführung abgebrochen. Der Timeout-Fehler ist dann über die beiden Ausgänge `bError` und `nErrorId` zu erkennen.

Arbeitsstatus

Der Ausgang `bBusy` gibt an, ob der Funktionsblock beschäftigt ist. Wenn `bBusy = TRUE` kann kein neuer Befehl ausgelöst werden.

Mittels einer geeigneten `IF`-Abfrage kann festgestellt werden, ob die Ausführung des Funktionsblocks abgeschlossen ist:

```
IF NOT fbReadOrWrite.bBusy THEN
    // Function block execution is complete or error occurred.
    // Next write-command can be issued.
END_IF
```

Fehlerstatus

Der Fehlerstatus wird angegeben, indem `bError` binär auf einen Fehler hinweist und `nErrorId` ggfs. den Fehlercode (ADS Return Code [▶ 2966]) ausgibt. Der Fehlerstatus wird zurückgesetzt, sobald ein neuer Befehl ausgelöst wird. Der Fehlercode ist `PENDING (16#71E)`, wenn der Funktionsblock beschäftigt ist (`bBusy = TRUE`).

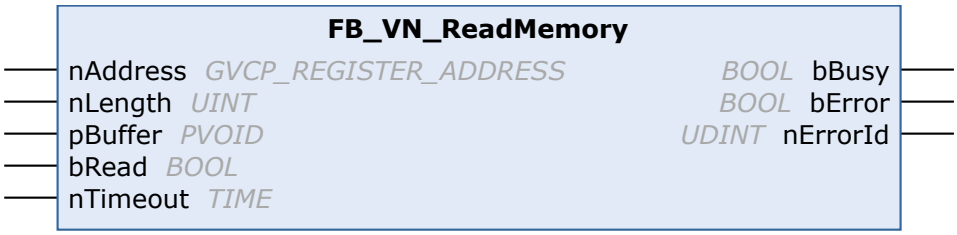
Der Hexadezimalwert des Fehlercodes kann wie folgt extrahiert werden:

```
IF fbReadOrWrite.bError THEN
    // Show relevant error code for debugging purposes:
    nErrorCode := fbReadOrWrite.nErrorId AND 16#FFF;
END_IF
```

Beispiele

- Lesen/Schreiben eines Registers [▶ 2919]

6.1.5.2.1 FB_VN_ReadMemory



This FB reads consecutive memory locations from the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before)

Syntax

Definition:

```
FUNCTION_BLOCK FB_VN_ReadMemory
VAR_INPUT
    nAddress                : GVCP_REGISTER_ADDRESS;
    nLength                 : UINT;
    pBuffer                 : PVOID;
    bRead                   : BOOL;
    nTimeout                : TIME;
END_VAR
VAR_OUTPUT
    bBusy                   : BOOL;
    bError                  : BOOL;
    nErrorId                : UDINT;
END_VAR
VAR
    oidITcVnGevImageProvider : OTCID;
END_VAR
```

Inputs

Name	Type	Default	Description
nAddress	GVCP_REGISTER_ADDRESS [▶ 171]		The address of the first byte to read from the camera (must be 32-bit aligned)
nLength	UINT		The number of bytes to read, starting with nAddress (must be a multiple of 4, not more than 536)
pBuffer	PVOID		Pointer to the buffer where the read memory content is written to (The buffer must have a size of at least nLength bytes!)
bRead	BOOL		Reading the memory is triggered by a rising edge at this input.
nTimeout	TIME	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	BOOL	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrorId'. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	UDINT	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Weiterführende Informationen

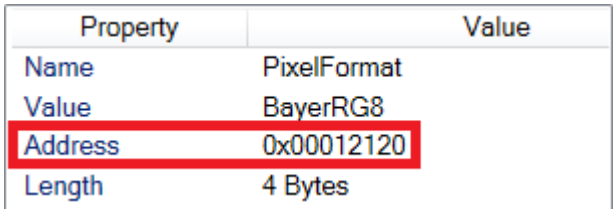
Offener Kamera-Control-Channel nötig!

Um diesen Funktionsblock benutzen zu können, muss der Control Channel der GigE Vision Kamera offen sein. Zudem können einige Kamera-Parameter nur geändert werden, wenn die Bild-Akquisition gestoppt ist. Daher wird empfohlen, die Kamera zuvor in den State TCVN_CS_OPENED zu bringen.

Parameter

nAddress

Die Register-Adresse identifiziert das zu adressierende Register der Kamera. Sie ist vom Typ `GVCP_REGISTER_ADDRESS`, welcher ein Alias auf den Typ `UDINT` ist. Damit haben alle Register-Adressen eine Größe von 4 Bytes. Die Register-Adresse eines Kamera-Parameters können Sie an entsprechender Stelle im Konfigurationsbaum [▶ 83] der GigE Vision Kamera Instanz nachsehen:



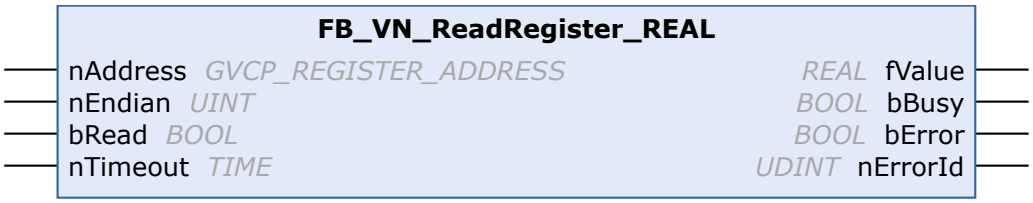
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.2.2 FB_VN_ReadRegister_REAL



This FB reads a REAL register from the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before)

Syntax

Definition:

```
FUNCTION_BLOCK FB_VN_ReadRegister_REAL
VAR_INPUT
    nAddress                : GVCP_REGISTER_ADDRESS;
    nEndian                 : UINT;
    bRead                   : BOOL;
    nTimeout                : TIME;
END_VAR
VAR_OUTPUT
    fValue                  : REAL;
    bBusy                   : BOOL;
    bError                  : BOOL;
    nErrorId                : UDINT;
END_VAR
VAR
    oidITcVnGevImageProvider : OTCID;
END_VAR
```

Inputs

Name	Type	Default	Description
nAddress	GVCP_REGISTER_ADDRESS [▶ 171]		The address of the register that should be read
nEndian	UINT		OPTIONAL: The endianness of the register. 0 = Big, 1 = Little. Default: 0
bRead	BOOL		Reading the register is triggered by a rising edge at this input.
nTimeout	TIME	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
fValue	REAL	The read value
bBusy	BOOL	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	UDINT	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Weiterführende Informationen

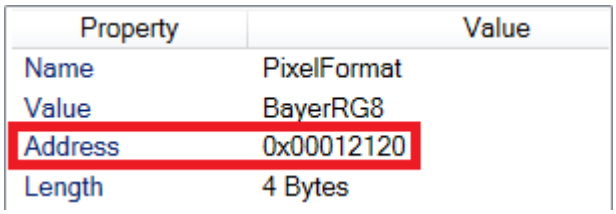
Offener Kamera-Control-Channel nötig!

Um diesen Funktionsblock benutzen zu können, muss der Control Channel der GigE Vision Kamera offen sein. Zudem können einige Kamera-Parameter nur geändert werden, wenn die Bild-Akquisition gestoppt ist. Daher wird empfohlen, die Kamera zuvor in den State TCVN_CS_OPENED zu bringen.

Parameter

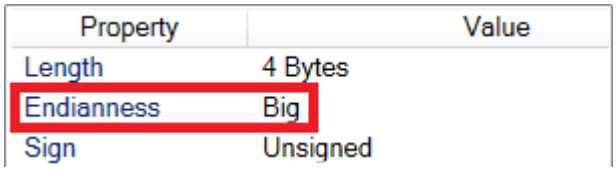
nAddress

Die Register-Adresse identifiziert das zu adressierende Register der Kamera. Sie ist vom Typ `GVCP_REGISTER_ADDRESS`, welcher ein Alias auf den Typ `UDINT` ist. Damit haben alle Register-Adressen eine Größe von 4 Bytes. Die Register-Adresse eines Kamera-Parameters können Sie an entsprechender Stelle im Konfigurationsbaum [▶ 83] der GigE Vision Kamera Instanz nachsehen:



nEndian

Die Endianess beschreibt die Reihenfolge, in der die Bytes eines Kamera-Registers angeordnet werden. Diese muss anwendungstechnisch nicht weiter beachtet werden und kann als Eigenschaft des jeweiligen Parameters vom Konfigurationsbaum der GigE Vision Kamera Instanz übernommen werden:



Ein Big Endian wird dabei durch `nEndian := 0` und ein Small Endian durch `nEndian := 1` dargestellt. Big Endian ist Standard.

Beispiele

- Lesen/Schreiben eines Registers [▶ 2919]

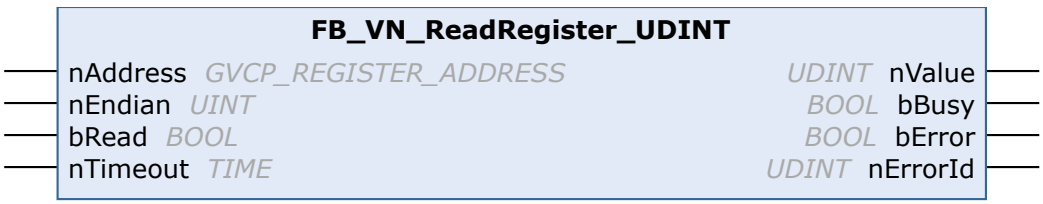
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.2.3 FB_VN_ReadRegister_UDINT



This FB reads a DINT register from the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before)

Syntax

Definition:

```
FUNCTION_BLOCK FB_VN_ReadRegister_UDINT
VAR_INPUT
    nAddress                : GVCP_REGISTER_ADDRESS;
    nEndian                 : UINT;
```

```
    bRead                    : BOOL;
    nTimeout                 : TIME;
END_VAR
VAR_OUTPUT
    nValue                   : UDINT;
    bBusy                    : BOOL;
    bError                   : BOOL;
    nErrorId                 : UDINT;
END_VAR
VAR
    oidITcVnGevImageProvider : OTCID;
END_VAR
```

Inputs

Name	Type	Default	Description
nAddress	GVCP_REGISTER_ADDRESS [▶ 171]		The address of the register that should be read
nEndian	UINT		OPTIONAL: The endianness of the register. 0 = Big, 1 = Little. Default: 0
bRead	BOOL		Reading the register is triggered by a rising edge at this input.
nTimeout	TIME	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
nValue	UDINT	The read value
bBusy	BOOL	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	UDINT	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Weiterführende Informationen

Offener Kamera-Control-Channel nötig!

Um diesen Funktionsblock benutzen zu können, muss der Control Channel der GigE Vision Kamera offen sein. Zudem können einige Kamera-Parameter nur geändert werden, wenn die Bild-Akquisition gestoppt ist. Daher wird empfohlen, die Kamera zuvor in den State TCVN_CS_OPENED zu bringen.

Parameter

nAddress

Die Register-Adresse identifiziert das zu adressierende Register der Kamera. Sie ist vom Typ `GVCP_REGISTER_ADDRESS`, welcher ein Alias auf den Typ `UDINT` ist. Damit haben alle Register-Adressen eine Größe von 4 Bytes. Die Register-Adresse eines Kamera-Parameters können Sie an entsprechender Stelle im Konfigurationsbaum [▶ 83] der GigE Vision Kamera Instanz nachsehen:

Property	Value
Name	PixelFormat
Value	BayerRG8
Address	0x00012120
Length	4 Bytes

nEndian

Die Endianess beschreibt die Reihenfolge, in der die Bytes eines Kamera-Registers angeordnet werden. Diese muss anwendungstechnisch nicht weiter beachtet werden und kann als Eigenschaft des jeweiligen Parameters vom Konfigurationsbaum der GigE Vision Kamera Instanz übernommen werden:

Property	Value
Length	4 Bytes
Endianness	Big
Sign	Unsigned

Ein Big Endian wird dabei durch `nEndian := 0` und ein Small Endian durch `nEndian := 1` dargestellt. Big Endian ist Standard.

Beispiele

- Lesen/Schreiben eines Registers [▶ 2919]

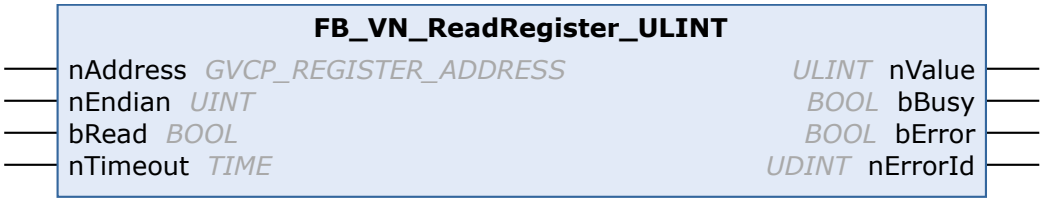
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.2.4 FB_VN_ReadRegister_ULINT



This FB reads a ULINT register from the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before)

Syntax

Definition:

```
FUNCTION_BLOCK FB_VN_ReadRegister_ULINT
VAR_INPUT
    nAddress                : GVCP_REGISTER_ADDRESS;
    nEndian                 : UINT;
    bRead                   : BOOL;
    nTimeout                : TIME;
END_VAR
VAR_OUTPUT
    nValue                  : ULINT;
    bBusy                   : BOOL;
    bError                  : BOOL;
    nErrorId                : UDINT;
END_VAR
```

```
VAR
    oidITcVnGevImageProvider : OTCID;
END_VAR
```

Inputs

Name	Type	Default	Description
nAddress	GVCP_REGISTER_ADDRESS [▶ 171]		The address of the register that should be read
nEndian	UINT		OPTIONAL: The endianness of the register. 0 = Big, 1 = Little. Default: 0
bRead	BOOL		Reading the register is triggered by a rising edge at this input.
nTimeout	TIME	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
nValue	ULINT	The read value
bBusy	BOOL	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	UDINT	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Weiterführende Informationen

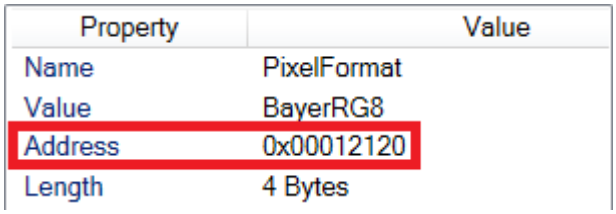
Offener Kamera-Control-Channel nötig!

Um diesen Funktionsblock benutzen zu können, muss der Control Channel der GigE Vision Kamera offen sein. Zudem können einige Kamera-Parameter nur geändert werden, wenn die Bild-Akquisition gestoppt ist. Daher wird empfohlen, die Kamera zuvor in den State TCVN_CS_OPENED zu bringen.

Parameter

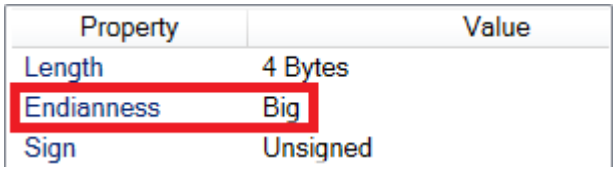
nAddress

Die Register-Adresse identifiziert das zu adressierende Register der Kamera. Sie ist vom Typ `GVCP_REGISTER_ADDRESS`, welcher ein Alias auf den Typ `UDINT` ist. Damit haben alle Register-Adressen eine Größe von 4 Bytes. Die Register-Adresse eines Kamera-Parameters können Sie an entsprechender Stelle im Konfigurationsbaum [▶ 83] der GigE Vision Kamera Instanz nachsehen:



nEndian

Die Endianess beschreibt die Reihenfolge, in der die Bytes eines Kamera-Registers angeordnet werden. Diese muss anwendungstechnisch nicht weiter beachtet werden und kann als Eigenschaft des jeweiligen Parameters vom Konfigurationsbaum der GigE Vision Kamera Instanz übernommen werden:



Ein Big Endian wird dabei durch `nEndian := 0` und ein Small Endian durch `nEndian := 1` dargestellt. Big Endian ist Standard.

Beispiele

Lesen/Schreiben eines Registers [▶ 2919]

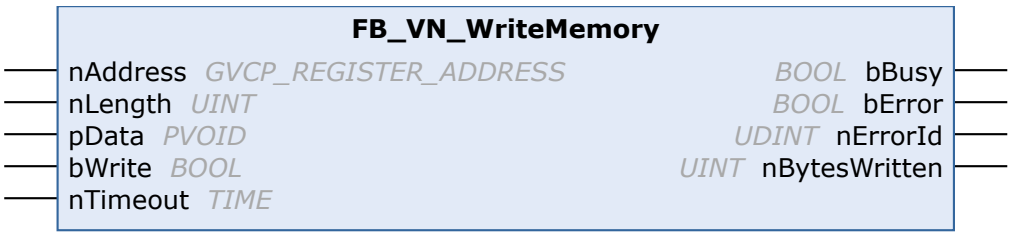
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.2.5 FB_VN_WriteMemory



This FB writes data to consecutive memory locations on the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before)

Syntax

Definition:

```
FUNCTION_BLOCK FB_VN_WriteMemory
VAR_INPUT
    nAddress                : GVCP_REGISTER_ADDRESS;
    nLength                 : UINT;
    pData                   : PVOID;
    bWrite                  : BOOL;
    nTimeout                : TIME;
END_VAR
VAR_OUTPUT
    bBusy                   : BOOL;
    bError                  : BOOL;
    nErrorId                : UDINT;
    nBytesWritten           : UINT;
END_VAR
VAR
    oidITcVnGevImageProvider : OTCID;
END_VAR
```

Inputs

Name	Type	Default	Description
nAddress	GVCP_REGISTER_ADDRESS [▶ 171]		The address of the first byte to write to the camera (must be 32-bit aligned)
nLength	UINT		The number of bytes to write, starting at nAddress (must be a multiple of 4, not more than 536)
pData	PVOID		Pointer to the data that should be written to the camera (must have a size of at least nLength bytes!)
bWrite	BOOL		Writing the memory is triggered by a rising edge at this input.
nTimeout	TIME	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	BOOL	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	UDINT	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.
nBytesWritten	UINT	The number of bytes that have been successfully written to the camera.

Weiterführende Informationen

Offener Kamera-Control-Channel nötig!

Um diesen Funktionsblock benutzen zu können, muss der Control Channel der GigE Vision Kamera offen sein. Zudem können einige Kamera-Parameter nur geändert werden, wenn die Bild-Akquisition gestoppt ist. Daher wird empfohlen, die Kamera zuvor in den State TCVN_CS_OPENED zu bringen.

Parameter

nAddress

Die Register-Adresse identifiziert das zu adressierende Register der Kamera. Sie ist vom Typ `GVCP_REGISTER_ADDRESS`, welcher ein Alias auf den Typ `UDINT` ist. Damit haben alle Register-Adressen eine Größe von 4 Bytes. Die Register-Adresse eines Kamera-Parameters können Sie an entsprechender Stelle im Konfigurationsbaum [▶ 83] der GigE Vision Kamera Instanz nachsehen:

Property	Value
Name	PixelFormat
Value	BayerRG8
Address	0x00012120
Length	4 Bytes

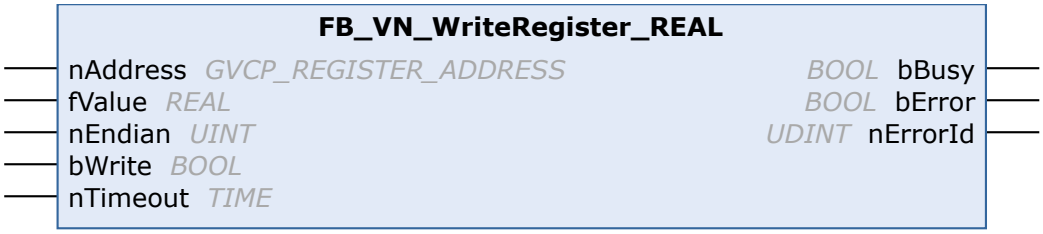
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.2.6 FB_VN_WriteRegister_REAL



This FB writes a REAL value into a register on the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before)

Syntax

Definition:

```
FUNCTION_BLOCK FB_VN_WriteRegister_REAL
VAR_INPUT
    nAddress                : GVCP_REGISTER_ADDRESS;
    fValue                  : REAL;
    nEndian                 : UINT;
    bWrite                  : BOOL;
    nTimeout                : TIME;
END_VAR
VAR_OUTPUT
    bBusy                   : BOOL;
    bError                  : BOOL;
    nErrorId                : UDINT;
END_VAR
VAR
    oidITcVnGevImageProvider : OTCID;
END_VAR
```

Inputs

Name	Type	Default	Description
nAddress	GVCP_REGISTER_ADDRESS [▶ 171]		The address of the register that should be written
fValue	REAL		The value to write
nEndian	UINT		OPTIONAL: The endianness of the register. 0 = Big, 1 = Little. Default: 0
bWrite	BOOL		Writing the register is triggered by a rising edge at this input.
nTimeout	TIME	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	BOOL	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	UDINT	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Weiterführende Informationen

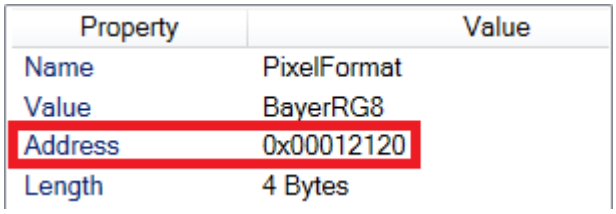
Offener Kamera-Control-Channel nötig!

Um diesen Funktionsblock benutzen zu können, muss der Control Channel der GigE Vision Kamera offen sein. Zudem können einige Kamera-Parameter nur geändert werden, wenn die Bild-Akquisition gestoppt ist. Daher wird empfohlen, die Kamera zuvor in den State TCVN_CS_OPENED zu bringen.

Parameter

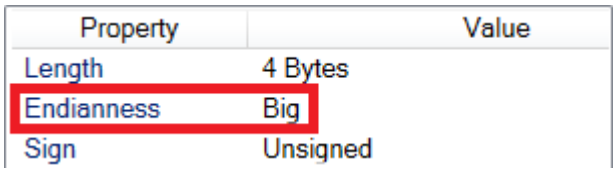
nAddress

Die Register-Adresse identifiziert das zu adressierende Register der Kamera. Sie ist vom Typ `GVCP_REGISTER_ADDRESS`, welcher ein Alias auf den Typ `UDINT` ist. Damit haben alle Register-Adressen eine Größe von 4 Bytes. Die Register-Adresse eines Kamera-Parameters können Sie an entsprechender Stelle im Konfigurationsbaum [▶ 83] der GigE Vision Kamera Instanz nachsehen:



nEndian

Die Endianess beschreibt die Reihenfolge, in der die Bytes eines Kamera-Registers angeordnet werden. Diese muss anwendungstechnisch nicht weiter beachtet werden und kann als Eigenschaft des jeweiligen Parameters vom Konfigurationsbaum der GigE Vision Kamera Instanz übernommen werden:



Ein Big Endian wird dabei durch `nEndian := 0` und ein Small Endian durch `nEndian := 1` dargestellt. Big Endian ist Standard.

Beispiele

- Lesen/Schreiben eines Registers [▶ 2919]

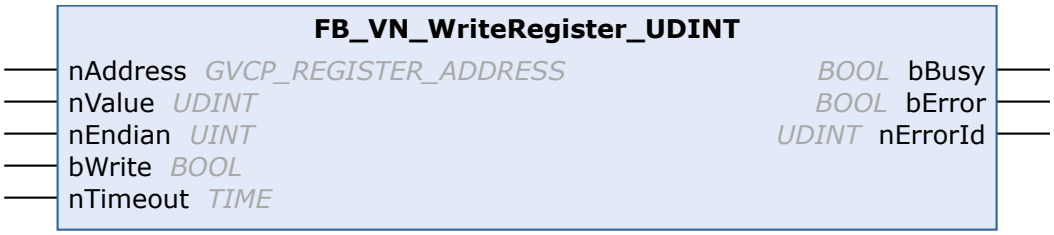
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.2.7 FB_VN_WriteRegister_UDINT



This FB writes a DINT value into a register on the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before)

Syntax

Definition:

```
FUNCTION_BLOCK FB_VN_WriteRegister_UDINT
VAR_INPUT
    nAddress                : GVCP_REGISTER_ADDRESS;
    nValue                  : UDINT;
    nEndian                 : UINT;
    bWrite                  : BOOL;
    nTimeout                : TIME;
END_VAR
VAR_OUTPUT
    bBusy                   : BOOL;
    bError                  : BOOL;
    nErrorId                : UDINT;
END_VAR
VAR
    oidITcVnGevImageProvider : OTCID;
END_VAR
```

Inputs

Name	Type	Default	Description
nAddress	GVCP_REGISTER_ADDRESS [▶ 171]		The address of the register that should be written
nValue	UDINT		The value to write
nEndian	UINT		OPTIONAL: The endianness of the register. 0 = Big, 1 = Little. Default: 0
bWrite	BOOL		Writing the register is triggered by a rising edge at this input.
nTimeout	TIME	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	BOOL	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	UDINT	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Weiterführende Informationen

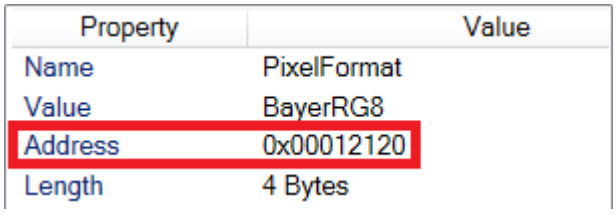
Offener Kamera-Control-Channel nötig!

Um diesen Funktionsblock benutzen zu können, muss der Control Channel der GigE Vision Kamera offen sein. Zudem können einige Kamera-Parameter nur geändert werden, wenn die Bild-Akquisition gestoppt ist. Daher wird empfohlen, die Kamera zuvor in den State TCVN_CS_OPENED zu bringen.

Parameter

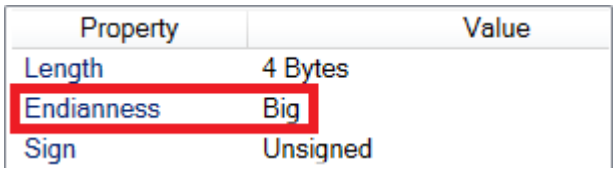
nAddress

Die Register-Adresse identifiziert das zu adressierende Register der Kamera. Sie ist vom Typ `GVCP_REGISTER_ADDRESS`, welcher ein Alias auf den Typ `UDINT` ist. Damit haben alle Register-Adressen eine Größe von 4 Bytes. Die Register-Adresse eines Kamera-Parameters können Sie an entsprechender Stelle im Konfigurationsbaum [▶ 83] der GigE Vision Kamera Instanz nachsehen:



nEndian

Die Endianess beschreibt die Reihenfolge, in der die Bytes eines Kamera-Registers angeordnet werden. Diese muss anwendungstechnisch nicht weiter beachtet werden und kann als Eigenschaft des jeweiligen Parameters vom Konfigurationsbaum der GigE Vision Kamera Instanz übernommen werden:



Ein Big Endian wird dabei durch `nEndian := 0` und ein Small Endian durch `nEndian := 1` dargestellt. Big Endian ist Standard.

Beispiele

- Lesen/Schreiben eines Registers [▶ 2919]

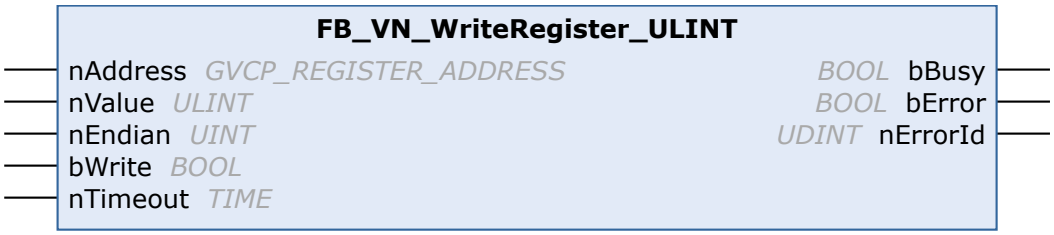
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.2.8 FB_VN_WriteRegister_ULINT



This FB writes a ULINT value into a register on the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before)

Syntax

Definition:

```
FUNCTION_BLOCK FB_VN_WriteRegister_ULINT
VAR_INPUT
    nAddress                : GVCP_REGISTER_ADDRESS;
    nValue                  : ULINT;
    nEndian                 : UINT;
    bWrite                  : BOOL;
    nTimeout                : TIME;
END_VAR
VAR_OUTPUT
    bBusy                   : BOOL;
    bError                  : BOOL;
    nErrorId                : UDINT;
END_VAR
VAR
    oidITcVnGevImageProvider : OTCID;
END_VAR
```

Inputs

Name	Type	Default	Description
nAddress	GVCP_REGISTER_ADDRESS [▶ 171]		The address of the register that should be written
nValue	ULINT		The value to write
nEndian	UINT		OPTIONAL: The endianness of the register. 0 = Big, 1 = Little. Default: 0
bWrite	BOOL		Writing the register is triggered by a rising edge at this input.
nTimeout	TIME	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	BOOL	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	UDINT	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Weiterführende Informationen

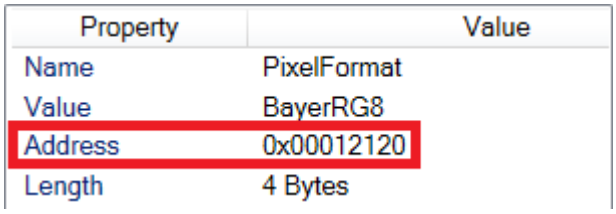
Offener Kamera-Control-Channel nötig!

Um diesen Funktionsblock benutzen zu können, muss der Control Channel der GigE Vision Kamera offen sein. Zudem können einige Kamera-Parameter nur geändert werden, wenn die Bild-Akquisition gestoppt ist. Daher wird empfohlen, die Kamera zuvor in den State TCVN_CS_OPENED zu bringen.

Parameter

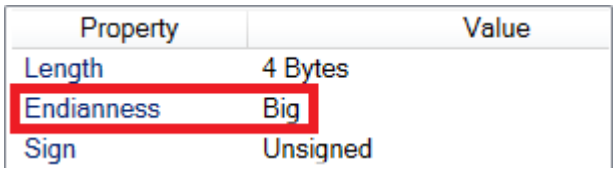
nAddress

Die Register-Adresse identifiziert das zu adressierende Register der Kamera. Sie ist vom Typ `GVCP_REGISTER_ADDRESS`, welcher ein Alias auf den Typ `UDINT` ist. Damit haben alle Register-Adressen eine Größe von 4 Bytes. Die Register-Adresse eines Kamera-Parameters können Sie an entsprechender Stelle im Konfigurationsbaum [▶ 83] der GigE Vision Kamera Instanz nachsehen:



nEndian

Die Endianess beschreibt die Reihenfolge, in der die Bytes eines Kamera-Registers angeordnet werden. Diese muss anwendungstechnisch nicht weiter beachtet werden und kann als Eigenschaft des jeweiligen Parameters vom Konfigurationsbaum der GigE Vision Kamera Instanz übernommen werden:



Ein Big Endian wird dabei durch `nEndian := 0` und ein Small Endian durch `nEndian := 1` dargestellt. Big Endian ist Standard.

Beispiele

Lesen/Schreiben eines Registers [▶ 2919]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.3 File Access

Diese Gruppe enthält Funktionsblöcke zum Lesen und Schreiben von Daten auf dem Dateisystem.

Funktionsblöcke

Bilder

- FB_VN_ReadImage [▶ 1648]
- FB_VN_WriteImage [▶ 1656]

Container

- FB_VN_ReadContainer [▶ 1646]
- FB_VN_WriteContainer [▶ 1654]

Machine learning Modelle

- FB_VN_ReadMlModel [▶ 1650]
- FB_VN_WriteMlModel [▶ 1658]

Kalibrationsmuster

- FB_VN_ReadCalibrationPattern [▶ 1642]
- FB_VN_ReadCalibrationResult [▶ 1645]
- FB_VN_WriteCalibrationResult [▶ 1653]

Funktionsinitialisierung

- FB_VN_InitializeFunction [▶ 1640]

Neural Network

- FB_VN_ReadNeuralNetwork [▶ 1660]

Allgemeines

Diese Funktionsblöcke arbeiten asynchron; die Zeit von der ersten Ausführung bis zum gewünschten Ergebnis erstreckt sich über mehrere Zyklen. Die tatsächliche Dauer jedes Lese-/Schreibvorgangs hängt von verschiedenen Faktoren ab, z.B. von der Größe des Bildes und dem Verhalten des Betriebssystems. Die folgenden Parameter sind bei allen Funktionsblöcken gleich und sind daher einmal hier zentral beschrieben.

Parameter

Dateipfad

Der Dateipfad `sFilePath` legt fest, wo eine Datei gespeichert werden soll bzw. welche Datei geladen werden soll. Es können absolute und relative Pfade verwendet werden. Bei relativen Pfaden werden die Standardpfade, die in der Service Configuration [▶ 68] festgelegt sind, als Basis verwendet.

Pfadlänge

Der Typ des `sFilePath` ist als STRING angegeben, es können aber Pfade mit einer Länge bis zu 255 Zeichen übergeben werden.

Der Dateipfad wird bei einer steigenden Flanke des Schreib-Befehls `bWrite` intern gespeichert, sodass Änderungen des Pfads in späteren Zyklen keine Auswirkung haben.

Funktion ausführen

Die Trigger für Schreib- und Lese-Befehle sind `bWrite` und `bRead`. Sie werden durch eine steigende Flanke (Wechsel von `FALSE` nach `TRUE`) ausgelöst. Dabei werden die relevanten Eingangsparameter intern gespeichert, sodass eine Änderung der Parameter während der Ausführung keinen Einfluss hat. Ebenso hat eine fallende Flanke der Trigger keine Auswirkung.

Funktionsblöcke aufrufen

Beachten Sie, dass das Beschreiben von Parametern bei Funktionsblöcken nur eine Auswirkung hat, wenn dieser auch aufgerufen wird.

Timeout

Um in unbekannten Fehlerfällen nicht ewig auf die Ausführung des Funktionsblocks zu warten, kann mit `nTimeout` die maximal erlaubte Ausführungszeit festgelegt werden. Wenn diese Zeit ohne erfolgreiches Ergebnis nach Auslösen des Befehlstriggers überschritten wird, wird die Ausführung abgebrochen. Der Timeout-Fehler ist dann über die beiden Ausgänge `bError` und `nErrorId` zu erkennen.

Arbeitsstatus

Der Ausgang `bBusy` gibt an, ob der Funktionsblock beschäftigt ist. Wenn `bBusy = TRUE` kann kein neuer Befehl ausgelöst werden.

Mittels einer geeigneten `IF`-Abfrage kann festgestellt werden, ob die Ausführung des Funktionsblocks abgeschlossen ist:

```
IF NOT fbReadOrWrite.bBusy THEN
    // Function block execution is complete or error occurred.
    // Next write-command can be issued.
END_IF
```

Fehlerstatus

Der Fehlerstatus wird angegeben, indem `bError` binär auf einen Fehler hinweist und `nErrorId` ggfs. den Fehlercode (ADS Return Code [▶ 2966]) ausgibt. Der Fehlerstatus wird zurückgesetzt, sobald ein neuer Befehl ausgelöst wird. Der Fehlercode ist `PENDING (16#71E)`, wenn der Funktionsblock beschäftigt ist (`bBusy = TRUE`).

Der Hexadezimalwert des Fehlercodes kann wie folgt extrahiert werden:

```
IF fbReadOrWrite.bError THEN
    // Show relevant error code for debugging purposes:
    nErrorCode := fbReadOrWrite.nErrorId AND 16#FFF;
END_IF
```

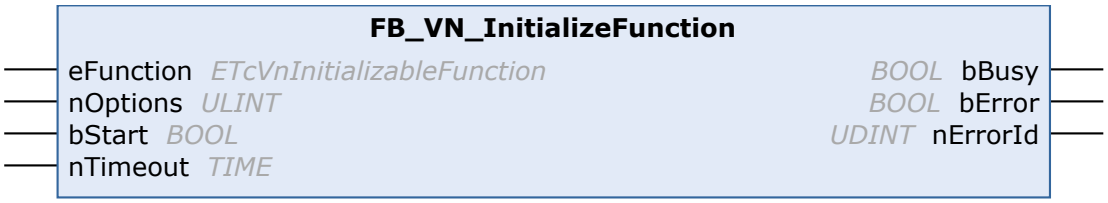
Die häufigsten Fehlercodes sind:

Hex	Dec	Name	Bedeutung
16#006	6	TARGETPORTNOTFOUND	ADS-Server nicht erreichbar. Tritt im Wesentlichen auf, wenn der TwinCAT Vision Service nicht gestartet ist. Siehe Vision Service startet nicht [▶ 2951].
16#700	1792	ERROR	Allgemeiner Fehler, wenn das Speichern oder Laden fehlschlägt. Dieser weist meistens darauf hin, dass das angegebene Dateiformat nicht unterstützt wird oder, dass der Dateipfad nicht erreichbar ist. Dieser Fehler tritt im Regelfall im TwinCAT Vision Service [▶ 68] auf.
16#70B	1803	INVALIDPARM	Invalider Parameter. Dies weist meistens darauf hin, dass der Interface Pointer [▶ 146] nicht valide ist.
16#719	1817	TIMEOUT	Timeout. Das Laden oder Speichern hat die maximale Ausführungszeit nTimeout überschritten. Generell tritt dies hauptsächlich bei starker Auslastung von ADS auf.
16#71E	1822	PENDING	Kein wirklicher Fehler, daher ist hierbei auch der Ausgang bError nicht gesetzt. Dieser Code signalisiert, dass der Funktionsblock beschäftigt ist. Er tritt parallel zum Ausgang bBusy auf, falls während Beschäftigtseins des Funktionsblocks eine neue steigende Flanke registriert wird.

Beispiele

- Speichern von Bildern aus der SPS [▶ 2921]

6.1.5.3.1 FB_VN_InitializeFunction



This FB initializes selected functions.

Syntax

Definition:

```
FUNCTION_BLOCK FB_VN_InitializeFunction
VAR_INPUT
    eFunction : ETcVnInitializableFunction;
    nOptions  : ULINT;
    bStart    : BOOL;
    nTimeout  : TIME;
END_VAR
VAR_OUTPUT
    bBusy     : BOOL;
    bError    : BOOL;
    nErrorId  : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
eFunction	ETcVnInitializableFunction [▶ 217]		Selects the function to initialize
nOptions	ULINT	0	Selects initialization options for the function
bStart	BOOL		Function initialization is triggered by a rising edge at this input.
nTimeout	TIME	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	BOOL	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	UDINT	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Weiterführende Informationen

Der Funktionsblock `FB_VN_InitializeFunction` initialisiert die unter `eFunction` angegebene Funktion. Im Hintergrund werden die benötigten Daten asynchron in die SPS geladen.

Parameter

Einige Parameter sind zentral im Kapitel File Access Parameter [▶ 1639] beschrieben. Abweichend von der Beschreibung muss bei diesem Funktionsblock die zu initialisierende Funktion anstatt eines Dateipfads angegeben werden.

Funktion

Die zu initialisierende `eFunction`.

Optionen

Auswahl der Initialisierungsoptionen `nOptions` für die angegebene Funktion. Die möglichen Enum-Optionen sind in der Beschreibung der zu initialisierenden Funktion zu finden. Die Optionen können mit einem `OR` verkettet und so mit einem Aufruf des Funktionsblocks geladen werden.

Anwendung

Die Initialisierung der OCR-Funktion mit der Option zur Klassifizierung von Großbuchstaben sieht z. B. so aus:

```
fbInitOCR(
    eFunction   := TCVN_IF_OCR,
    nOptions    := ETcVnOcrModelType.TCVN_OMT_UCLETTERS,
    bStart      := TRUE,
    nTimeout    := T#500MS);

IF NOT fbInitOCR.bBusy AND NOT fbInitOCR.bError THEN
    // The function is initialized and can be used with the specified options
END_IF
```

Verwandte Funktionen

- Mit F_VN_CheckFunctionInitialization [▶ 1560] kann geprüft werden, ob eine Funktion mit einer entsprechenden Option initialisiert ist.
- F_VN_DeinitializeFunction [▶ 1568] bietet die Möglichkeit, eine Option von einer Funktion wieder zu de-initialisieren und so den Speicher freizugeben.

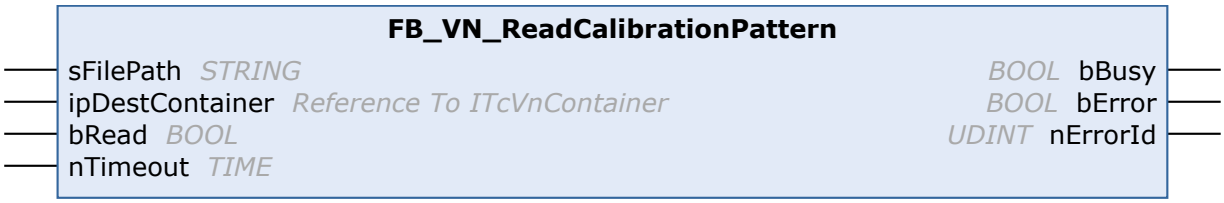
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.3.2 FB_VN_ReadCalibrationPattern



This FB reads calibration pattern reference points from an xml file on the targte pc.

Syntax

Definition:

```
FUNCTION_BLOCK FB_VN_ReadCalibrationPattern
VAR_INPUT
    sFilePath       : STRING;
    ipDestContainer : Reference To ITcVnContainer;
    bRead           : BOOL;
    nTimeout        : TIME;
END_VAR
VAR_OUTPUT
    bBusy           : BOOL;
    bError          : BOOL;
    nErrorId        : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
sFilePath	STRING		Full path of the file or relative path to the default directory on the target pc
ipDestContainer	Reference To ITcVnContainer [▶ 385]		Returns a container with the pattern points (ContainerType_Vector_TcVnPoint3_REAL)
bRead	BOOL		Reading the file is triggered by a rising edge at this input.
nTimeout	TIME	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	BOOL	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	UDINT	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Weiterführende Informationen

Der Funktionsblock `FB_VN_ReadCalibrationPattern` lädt die Beschreibung eines individuellen Kalibrationsmusters [▶ 129] asynchron aus einer Datei.

Laden von geänderten Containern

Beim Einladen der Container cached der Vision-Service die Daten intern. Ein Einladen von geänderten Containern ist somit im Standard nicht möglich. Passen Sie dazu die die Cache Einstellungen [▶ 68] an oder sehen Sie das Beispiel Serielles Laden von Bildern in die SPS [▶ 2922] für eine Alternative.

Parameter

Einige Parameter sind zentral in der Sektion File Access Parameter [▶ 1639] beschrieben.

Geladener Container (Rückgabewert)

Der geladene Container mit den Referenzpunkten des Kalibriermusters ist nach erfolgreichem Lesevorgang über den Interface Pointer `ipPatternReferencePoints` zu erreichen.

Anwendung

Das Laden der Referenzpunkte eines Kalibriermusters vom Pfad `C:\TcVision\CalibrationPatternReferencePoints.xml` sieht z. B. so aus:

```
fbReadCalibrationPattern(
    sFilePath       := 'C:\TcVision\CalibrationPatternReferencePoints.xml',
    ipDestContainer := ipPatternReferencePoints,
    bRead           := TRUE,
    nTimeout        := T#500MS);
```

```
IF NOT fbReadCalibrationPattern.bBusy AND NOT fbReadCalibrationPattern.bError THEN
    // reference points are accessible in ipPatternReferencePoints
END_IF
```

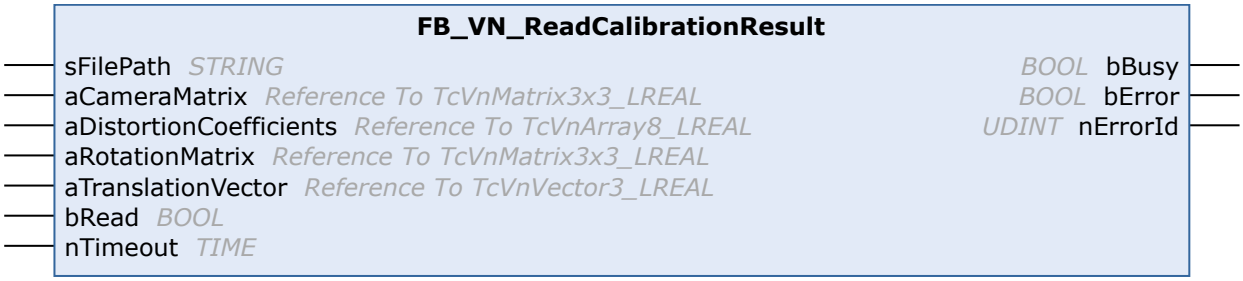
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.3.3 FB_VN_ReadCalibrationResult



This FB reads calibration results from a json file on the targte pc.

Syntax

Definition:

```
FUNCTION_BLOCK FB_VN_ReadCalibrationResult
VAR_INPUT
    sFilePath               : STRING;
    aCameraMatrix           : Reference To TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : Reference To TcVnArray8_LREAL;
    aRotationMatrix         : Reference To TcVnMatrix3x3_LREAL;
    aTranslationVector      : Reference To TcVnVector3_LREAL;
    bRead                   : BOOL;
    nTimeout                : TIME;
END_VAR
VAR_OUTPUT
    bBusy                   : BOOL;
    bError                  : BOOL;
    nErrorId                : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
sFilePath	STRING		Full path of the file or relative path to the default directory on the target pc
aCameraMatrix	Reference To TcVnMatrix3x3_LREAL [▶ 173]		Returns the camera matrix
aDistortionCoefficients	Reference To TcVnArray8_LREAL [▶ 173]		Returns the distortion coefficients
aRotationMatrix	Reference To TcVnMatrix3x3_LREAL [▶ 173]		Returns the rotation matrix
aTranslationVector	Reference To TcVnVector3_LREAL [▶ 173]		Returns the translation vector
bRead	BOOL		Reading the file is triggered by a rising edge at this input.
nTimeout	TIME	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	BOOL	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	UDINT	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Weiterführende Informationen

Der Funktionsblock `FB_VN_ReadCalibrationResult` lädt die Kalibration-Ergebnisse aus einer Datei asynchron in die SPS.

Laden von geänderten Kalibration-Ergebnissen.

Beim Einladen der Kalibration-Ergebnisse cached der Vision-Service die Daten intern. Ein Einladen von geänderten Ergebnissen ist somit im Standard nicht möglich. Passen Sie dazu die die Cache Einstellungen [▶ 68] an oder sehen Sie das Beispiel Serielles Laden von Bildern in die SPS [▶ 2922] für eine Alternative.

Parameter

Einige Parameter sind zentral in der Sektion File Access Parameter [▶ 1639] beschrieben.

Kalibration-Ergebnisse

Die geladenen Kalibration-Ergebnisse sind nach erfolgreichem Lesevorgang über die folgenden Variablen erreichbar:

```
// Calibration results
aCameraMatrix            : TcVnMatrix3x3_LREAL;
aDistortionCoefficients  : TcVnArray8_LREAL;
aRotationMatrix          : TcVnMatrix3x3_LREAL;
aTranslationVector       : TcVnVector3_LREAL;
```

Anwendung

Das Laden der Kalibration-Ergebnisse vom Pfad `C:\TcVision\ CalibrationResult.json` sieht z. B. so aus:

```
fbReadCalibrationResult(
        sFilePath                := 'C:\TcVision\ CalibrationResult.json',
        aCameraMatrix            := aCameraMatrix,
        aDistortionCoefficients  := aDistortionCoefficients,
        aRotationMatrix          := aRotationMatrix,
        aTranslationVector       := aTranslationVector,
        bRead                    := TRUE,
        nTimeout                 := T#500MS);

IF NOT fbReadCalibrationResult.bBusy AND NOT fbReadCalibrationResult.bError THEN
    // calibration results are accessible
END_IF
```

Verwandter Funktionsblock: FB_VN_WriteCalibrationResult [▶ 1652].

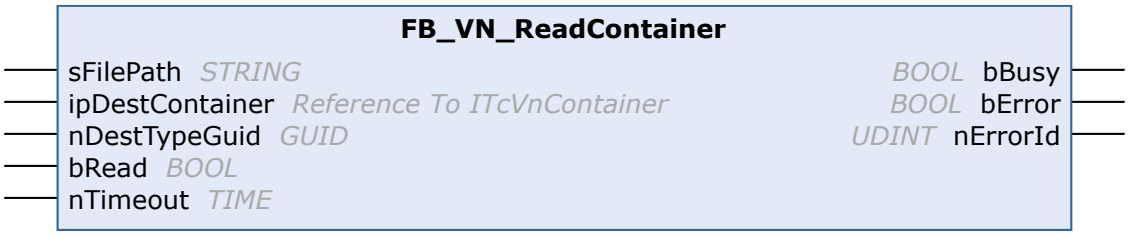
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.3.4 FB_VN_ReadContainer



This FB reads a container from an xml file on the target pc.

Syntax

Definition:

```
FUNCTION_BLOCK FB_VN_ReadContainer
VAR_INPUT
    sFilePath       : STRING;
    ipDestContainer : Reference To ITcVnContainer;
    nDestTypeGuid   : GUID;
    bRead           : BOOL;
    nTimeout        : TIME;
END_VAR
VAR_OUTPUT
    bBusy           : BOOL;
    bError          : BOOL;
    nErrorId        : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
sFilePath	STRING		Full path of the file or relative path to the default directory on the target pc
ipDestContainer	Reference To ITcVnContainer [▶ 385]		Returns a container with the loaded content
nDestTypeGuid	GUID		The type id of the container to return. If this parameter is not set, a suitable container type will be determined automatically (except for csv files).
bRead	BOOL		Reading the file is triggered by a rising edge at this input.
nTimeout	TIME	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	BOOL	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	UDINT	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Weiterführende Informationen

Der Funktionsblock `FB_VN_ReadContainer` lädt einen Container [▶ 163] aus einer Datei asynchron in die SPS.

Laden von geänderten Containern.

Beim Einladen der Container cached der Vision-Service die Daten intern. Ein Einladen von geänderten Containern ist somit im Standard nicht möglich. Passen Sie dazu die die Cache Einstellungen [▶ 68] an oder sehen Sie das Beispiel Serielles Laden von Bildern in die SPS [▶ 2922] für eine Alternative.

Parameter

Einige Parameter sind zentral in der Sektion File Access Parameter [▶ 1639] beschrieben.

Container-Typ

Der Container-Typ [▶ 171] des zu ladenden Containers kann optional angegeben werden.

Geladener Container (Rückgabewert)

Der geladene Container ist nach erfolgreichem Lesevorgang über den Interface Pointer `ipDestContainer` zu erreichen.

Anwendung

Das Laden eines Containers mit expliziter Typangabe `ContainerType_Vector_TcVnPoint2_DINT` vom Pfad `C:\TcVision\Container.xml` sieht z. B. so aus:

```
fbReadContainer(
    sFilePath       := 'C:\TcVision\Container.xml',
    ipDestContainer := ipContainer,
    nDestTypeGuid   := ContainerType_Vector_TcVnPoint2_DINT,
    bRead           := TRUE,
    nTimeout        := T#500MS);

IF NOT fbReadContainer.bBusy AND NOT fbReadContainer.bError THEN
    // container is accessible in ipContainer
END_IF
```

Verwandter Funktionsblock: FB_VN_WriteContainer [▶ 1654].

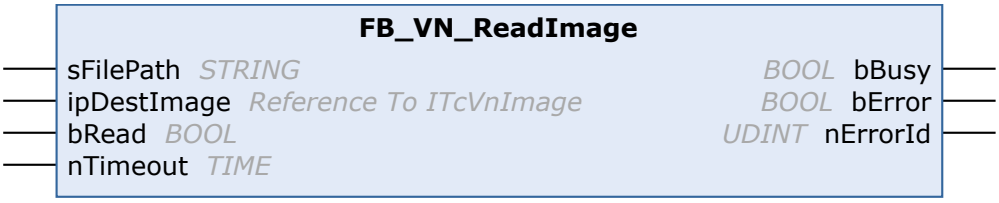
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.3.5 FB_VN_ReadImage



This FB reads an image from a file on the target pc.

Syntax

Definition:

```
FUNCTION_BLOCK FB_VN_ReadImage
VAR_INPUT
    sFilePath   : STRING;
    ipDestImage : Reference To ITcVnImage;
    bRead       : BOOL;
    nTimeout    : TIME;
END_VAR
VAR_OUTPUT
    bBusy       : BOOL;
    bError      : BOOL;
    nErrorId    : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
sFilePath	STRING		Full path of the file or relative path to the default image directory on the target pc
ipDestImage	Reference To ITcVnImage [▶ 426]		Returns an image with the loaded content
bRead	BOOL		Reading the file is triggered by a rising edge at this input.
nTimeout	TIME	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	BOOL	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in 'nErrorId'. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	UDINT	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Weiterführende Informationen

Der Funktionsblock `FB_VN_ReadImage` lädt ein Bild [▶ 159] aus einer Datei asynchron in die SPS.

Laden von geänderten Bildern

Beim Einladen der Bilder cached der Vision-Service die Daten intern. Ein Einladen von geänderten Bildern ist somit im Standard nicht möglich. Passen Sie dazu die die Cache Einstellungen [▶ 68] an oder sehen Sie das Beispiel Serielles Laden von Bildern in die SPS [▶ 2922] für eine Alternative.

Parameter

Einige Parameter sind zentral in der Sektion File Access Parameter [▶ 1639] beschrieben.

Zielbild (Rückgabewert)

Das geladene Bild ist nach erfolgreichem Lesevorgang über den Interface Pointer `ipDestImage` zu erreichen.

Anwendung

Das Laden eines Bilds vom Pfad `C:\TcVision\Image.bmp` sieht z. B. so aus:

```
fbReadImage(
    sFilePath   := 'C:\TcVision\Image.bmp',
    ipDestImage := ipImageIn,
    bRead       := TRUE,
    nTimeout    := T#500MS);

IF NOT fbReadImage.bBusy AND NOT fbReadImage.bError THEN
    // image is accessible in ipImageIn
END_IF
```

Verwandter Funktionsblock: FB_VN_WriteImage [▶ 1656] zum Speichern von Bildern.

Beispiele

- Serielles Laden von Bildern in die SPS [▶ 2922]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.3.6 FB_VN_ReadMlModel

This FB reads a machine learning model from a file on the target pc.

Syntax

Definition:

```
FUNCTION_BLOCK FB_VN_ReadMlModel
VAR_INPUT
    sFilePath     : STRING;
    ipDestMlModel : Reference To ITcVnMlModel;
    bRead         : BOOL;
    nTimeout      : TIME;
END_VAR
VAR_OUTPUT
    bBusy         : BOOL;
    bError        : BOOL;
    nErrorId      : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
sFilePath	STRING		Full path of the file or relative path to the default directory on the target pc
ipDestMlModel	Reference To ITcVnMlModel [▶ 435]		Returns the loaded machine learning model
bRead	BOOL		Reading the file is triggered by a rising edge at this input.
nTimeout	TIME	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	BOOL	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	UDINT	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Weiterführende Informationen

Der Funktionsblock `FB_VN_ReadMlModel` lädt ein ML-Modell [▶ 435] aus einer Datei asynchron in die SPS.

Laden von geänderten ML-Modellen

Beim Einladen des ML-Modells cached der Vision-Service die Daten intern. Ein Einladen von geänderten ML-Modellen ist somit im Standard nicht möglich. Passen Sie dazu die die Cache Einstellungen [▶ 68] an oder sehen Sie das Beispiel Serielles Laden von Bildern in die SPS [▶ 2922] für eine Alternative.

Parameter

Einige Parameter sind zentral im Kapitel File Access Parameter [▶ 1639] beschrieben.

Geladenes ML-Modell (Rückgabewert)

Das geladene `ITcVnMlModel` ist nach erfolgreichem Lesevorgang über den Interface Pointer `ipDestMlModel` zu erreichen.

Einen Sonderfall stellt die Verwendung als `ITcVnColorModel` dar, da nach dem Laden des Modells noch eine Umwandlung mit folgender Zeile erforderlich ist. Alternativ kann das `ITcVnMlModel` auch mit den entsprechenden Funktionen direkt weiterverwendet werden.

```
hr := ipReadMlModel.TcQueryInterface(IID_ITcVnColorModel, ADR(ipColorModel));
```

Anwendung

Das Laden eines ML-Modells vom Pfad `C:\TcVision\MlModel.vmlm` sieht z. B. so aus:

```
fbReadMlModel(
    sFilePath       := 'C:\TcVision\MlModel.vmlm',
    ipDestMlModel   := ipReadMlModel,
    bRead           := TRUE,
    nTimeout        := T#500MS);

IF NOT fbReadMlModel.bBusy AND NOT fbReadMlModel.bError THEN
    // Ml-Model is accessible in ipReadMlModel
END_IF
```

Verwandter Funktionsblock: FB_VN_WriteMlModel [▶ 1658].

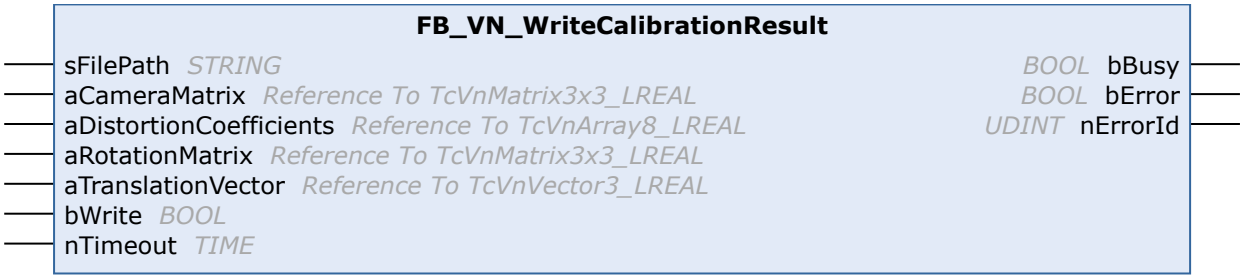
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.3.7 FB_VN_WriteCalibrationResult



This FB writes calibration results as a json file to the hard drive.

Syntax

Definition:

```
FUNCTION_BLOCK FB_VN_WriteCalibrationResult
VAR_INPUT
    sFilePath               : STRING;
    aCameraMatrix           : Reference To TcVnMatrix3x3_LREAL;
    aDistortionCoefficients : Reference To TcVnArray8_LREAL;
    aRotationMatrix         : Reference To TcVnMatrix3x3_LREAL;
    aTranslationVector      : Reference To TcVnVector3_LREAL;
    bWrite                  : BOOL;
    nTimeout                : TIME;
END_VAR
VAR_OUTPUT
    bBusy                   : BOOL;
    bError                  : BOOL;
    nErrorId                : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
sFilePath	STRING		Full path of the file or relative path to the default directory on the target pc (e.g. only a file name). If the string is empty, a file name will be generated (containing the current time).
aCameraMatrix	Reference To TcVnMatrix3x3_LREAL [▶ 173]		Returns the camera matrix
aDistortionCoefficients	Reference To TcVnArray8_LREAL [▶ 173]		Returns the distortion coefficients
aRotationMatrix	Reference To TcVnMatrix3x3_LREAL [▶ 173]		Returns the rotation matrix
aTranslationVector	Reference To TcVnVector3_LREAL [▶ 173]		Returns the translation vector
bWrite	BOOL		Writing the container is triggered by a rising edge at this input.
nTimeout	TIME	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	BOOL	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	UDINT	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Weiterführende Informationen

Der Funktionsblock `FB_VN_WriteCalibrationResult` speichert die Kalibration-Ergebnisse aus der SPS asynchron in einer Datei.

Parameter

Einige Parameter sind zentral im Kapitel File Access Parameter [▶ 1639] beschrieben.

Kalibration-Ergebnisse

Die folgenden Ergebnisse der Kalibrationsfunktionen [▶ 1101] werden unter dem angegebenen Dateipfad im `JSON`-Format gespeichert.

```
// Calibration results
aCameraMatrix            : TcVnMatrix3x3_LREAL;
aDistortionCoefficients  : TcVnArray8_LREAL;
aRotationMatrix          : TcVnMatrix3x3_LREAL;
aTranslationVector       : TcVnVector3_LREAL;
```

Dateipfad und Format

Wenn `sFilePath` leer ist, wird ein Dateiname mit der aktuellen Zeit generiert und die Endung `json` verwendet.

Anwendung

Das Speichern der Kalibration-Ergebnisse in die Datei `C:\TcVision\CalibrationResult.json` sieht z. B. so aus:

```
fbWriteCalibrationResult(
    sFilePath               := 'C:\TcVision\CalibrationResult.json',
    aCameraMatrix           := aCameraMatrix,
    aDistortionCoefficients := aDistortionCoefficients,
    aRotationMatrix         := aRotationMatrix,
    aTranslationVector      := aTranslationVector,
    bWrite                  := TRUE,
    nTimeout                := T#500MS);

IF NOT fbWriteCalibrationResult.bBusy AND NOT fbWriteCalibrationResult.bError THEN
    // Calibration result was written successfully to file
END_IF
```

Verwandter Funktionsblock: FB_VN_ReadCalibrationResult [▶ 1644].

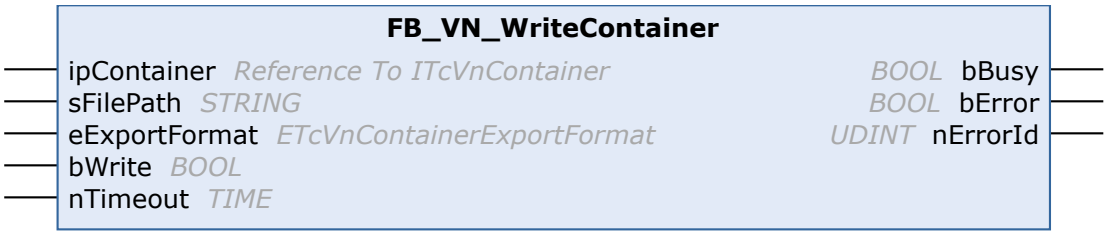
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.3.8 FB_VN_WriteContainer



This FB writes a container as an xml file to the hard drive.

Syntax

Definition:

```
FUNCTION_BLOCK FB_VN_WriteContainer
VAR_INPUT
    ipContainer   : Reference To ITcVnContainer;
    sFilePath     : STRING;
    eExportFormat : ETcVnContainerExportFormat;
    bWrite        : BOOL;
    nTimeout      : TIME;
END_VAR
VAR_OUTPUT
    bBusy         : BOOL;
```

```
    bError        : BOOL;
    nErrorId      : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
ipContainer	Reference To ITcVnContainer [▶ 385]		The container to write
sFilePath	STRING		Full path of the file or relative path to the default directory on the target pc (e.g. only a file name). If the string is empty, a file name will be generated (containing the current time).
eExportFormat	ETcVnContainerExportFormat [▶ 199]	TCVN_CEF_XML	Container export format (default is human readable xml)
bWrite	BOOL		Writing the container is triggered by a rising edge at this input.
nTimeout	TIME	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	BOOL	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	UDINT	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Weiterführende Informationen

Der Funktionsblock `FB_VN_WriteContainer` speichert einen Container [▶ 163] aus der SPS asynchron in einer Datei.

Parameter

Einige Parameter sind zentral in der Sektion File Access Parameter [▶ 1639] beschrieben.

Container

Der Container `ipContainer` wird unter dem angegebenen Dateipfad als XML-Datei gespeichert.

Export-Format

Das Export-Format `eExportFormat` legt über das Enum ETcVnContainerExportFormat [▶ 199] fest, ob der Container in menschenlesbarer Form oder serialisiert gespeichert werden soll:

- `TCVN_CEF_XML`
- `TCVN_CEF_XML_SERIALIZED`

- TCVN_CEF_CSV
- TCVN_CEF_BINARY

Die menschenlesbare Form (XML, CSV) hat den Vorteil, dass Einträge des Containers einfach per Hand überprüft und editiert werden können. Allerdings kann es zu Rundungsungenauigkeiten kommen. Das serialisierte Format hat den Vorteil, dass keine Rundungsungenauigkeiten auftreten. Jedoch ist das Format nicht gut lesbar oder editierbar. Das CSV-Format hat den Vorteil, dass die Daten extern in anderen Programmen einfach weiterverarbeitet werden können. Das Binärformat empfiehlt sich bei großen Datenmengen.

Anwendung

Das Speichern eines Containers im serialisierten XML-Format in die Datei `C:\TcVision\Container.xml` sieht z. B. so aus:

```
fbWriteContainer(
    ipContainer     := ipContainer,
    sFilePath       := 'C:\TcVision\Container.xml',
    eExportFormat   := TCVN_CEF_XML_SERIALIZED,
    bWrite          := TRUE,
    nTimeout        := T#500MS);

IF NOT fbWriteContainer.bBusy AND NOT fbWriteContainer.bError THEN
    // container was written successfully to file
END_IF
```

Verwandter Funktionsblock: FB_VN_ReadContainer [▶ 1646].

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.3.9 FB_VN_WriteImage

This FB writes an image to the hard drive.

Syntax

Definition:

```
FUNCTION_BLOCK FB_VN_WriteImage
VAR_INPUT
    ipImage   : Reference To ITcVnImage;
    sFilePath : STRING;
    bWrite    : BOOL;
    nTimeout  : TIME;
END_VAR
VAR_OUTPUT
    bBusy     : BOOL;
    bError    : BOOL;
    nErrorId  : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
ipImage	Reference To ITcVnImage [▶ 426]		The image to write
sFilePath	STRING		Full path of the file or relative path to the default directory on the target pc (e.g. only a file name). If the string is empty, a file name will be generated (containing the current time).
bWrite	BOOL		Writing the image is triggered by a rising edge at this input.
nTimeout	TIME	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	BOOL	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	UDINT	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Weiterführende Informationen

Der Funktionsblock `FB_VN_WriteImage` speichert ein Bild [▶ 159] aus der SPS asynchron in einer Datei.

Parameter

Einige Parameter sind zentral in der Sektion File Access Parameter [▶ 1639] beschrieben.

Eingangsbild

Das Format des Eingangsbilds `ipImage` muss mit dem im Dateipfad `sFilePath` festgelegten Dateiformat zusammenpassen.

Speichern bei steigender Flanke von bWrite

Das Bild wird bei einer steigenden Flanke des Schreib-Befehls intern kopiert. Somit haben Änderungen an dem Bild im folgenden Zyklus keine Auswirkungen. Es wird immer das Bild gespeichert, welches zum Zeitpunkt der steigenden Flanke von bWrite übergeben ist.

Dateipfad und Format

Der Dateipfad `sFilePath` legt fest, wo das Eingangsbild `ipImage` gespeichert wird und welches Bildformat dazu verwendet werden soll. Das Bildformat wird automatisch anhand der Dateiendung erkannt. Wenn kein Dateiname angegeben wird, wird dieser automatisch generiert und enthält die aktuelle Zeit. Wenn keine Dateiendung (z. B. `.jpg`) angegeben wird, wird automatisch die Endung `.bmp` und damit das BMP-Format verwendet.

Zum Speichern werden folgende Bildformate unterstützt.

- BMP (`.bmp`)
- PNG (`.png`)
- JPEG (`.jpeg` oder `.jpg`)
- TIF (`.tif`)
- TwinCAT Serialisierung (`.tcimg`)

HINWEIS

Fehlermöglichkeit

Achten Sie darauf, dass das gewählte Bildformat mit dem Format des tatsächlichen Eingangsbildes übereinstimmt. Anderenfalls können Fehler auftreten oder die gespeicherten Bilder fehlerhaft sein.

HINWEIS

16 Bit Bilder

Um 16-Bit-Bilder verlustfrei zu speichern, kommen nur das TIF-Format und die TwinCAT Serialisierung in Frage.

Anwendung

Das Speichern eines Bildes in die Datei `C:\TcVision\Image.bmp` sieht z. B. so aus:

```
fbWriteImage(
    ipImage     := ipImageRes,
    sFilePath   := 'C:\TcVision\Image.bmp',
    bWrite      := TRUE,
    nTimeout    := T#500MS);

IF NOT fbWriteImage.bBusy AND NOT fbWriteImage.bError THEN
    // ipImageRes was successfully written to given file path.
END_IF
```

Verwandter Funktionsblock: FB_VN_ReadImage [▶ 1648] zum Laden von Bildern.

Beispiele

- Speichern von Bildern aus der SPS [▶ 2921]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.3.10 FB_VN_WriteMlModel

This FB writes a machine learning model to the hard drive.

Syntax

Definition:

```
FUNCTION_BLOCK FB_VN_WriteMlModel
VAR_INPUT
    ipMlModel : Reference To ITcVnMlModel;
    sFilePath : STRING;
    bWrite    : BOOL;
    nTimeout  : TIME;
END_VAR
VAR_OUTPUT
    bBusy     : BOOL;
    bError    : BOOL;
    nErrorId  : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
ipMlModel	Reference To ITcVnMlModel [▶ 435]		The machine learning model to write
sFilePath	STRING		Full path of the file or relative path to the default directory on the target pc (e.g. only a file name). If the string is empty, a file name will be generated (containing the current time).
bWrite	BOOL		Writing the machine learning model is triggered by a rising edge at this input.
nTimeout	TIME	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	BOOL	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	UDINT	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Weiterführende Informationen

Der Funktionsblock `FB_VN_WriteMlModel` speichert ein ML-Modell [▶ 435] aus der SPS asynchron in einer Datei.

Parameter

Einige Parameter sind zentral im Kapitel File Access Parameter [▶ 1639] beschrieben.

ML-Modell

Das ML-Modell `ipMlModel` wird unter dem angegebenen Dateipfad im Binärformat gespeichert.

Einen Sonderfall stellt das Speichern eines `ITcVnColorModel` dar, da dieses zuerst noch mit der folgenden Zeile in ein ML-Modell gewandelt werden muss. Alternativ kann Color Model auch durch das `ITcVnMlModel` mit den entsprechenden Funktionen ersetzt werden.

```
hr := ipColorModel.TcQueryInterface(IID_ITcVnMlModel, ADR(ipWriteMlModel));
```

Dateipfad und Format

Wenn `sFilePath` leer ist, wird ein Dateiname mit der aktuellen Zeit generiert und die Endung vmlm verwendet.

Anwendung

Das Speichern eines ML-Models in die Datei `C:\TcVision\MlModel.vmlm` sieht z. B. so aus:

```
fbWriteMlModel(
    ipMlModel    := ipWriteMlModel,
    sFilePath    := 'C:\TcVision\MlModel.vmlm',
    bWrite       := TRUE,
    nTimeout     := T#500MS);

IF NOT fbWriteMlModel.bBusy AND NOT fbWriteMlModel.bError THEN
    // Ml-Model was written successfully to file
END_IF
```

Verwandter Funktionsblock: FB_VN_ReadMlModel [▶ 1650].

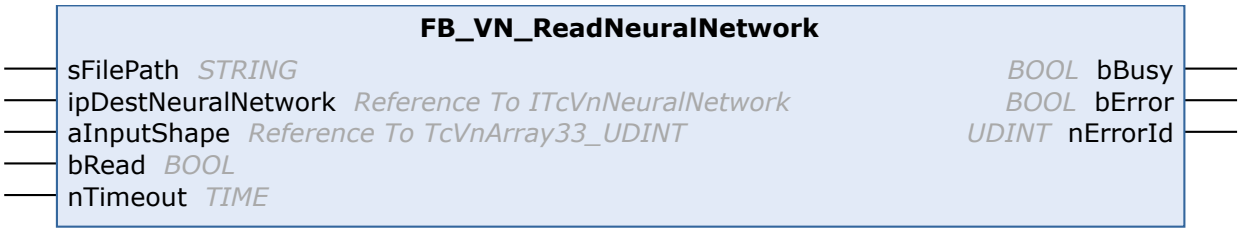
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.3.11 FB_VN_ReadNeuralNetwork



This FB reads a neural network from a file on the target pc.

Syntax

Definition:

```
FUNCTION_BLOCK FB_VN_ReadNeuralNetwork
VAR_INPUT
    sFilePath           : STRING;
    ipDestNeuralNetwork : Reference To ITcVnNeuralNetwork;
    aInputShape         : Reference To TcVnArray33_UDINT;
    bRead               : BOOL;
    nTimeout            : TIME;
END_VAR
VAR_OUTPUT
    bBusy               : BOOL;
    bError              : BOOL;
    nErrorId            : UDINT;
END_VAR
```

Inputs

Name	Type	Default	Description
sFilePath	STRING		Full path of the file or relative path to the default directory on the target pc
ipDestNeuralNetwork	Reference To ITcVnNeuralNetwork [▶ 445]		Returns the neural network
aInputShape	Reference To TcVnArray33_UDINT [▶ 173]		Optional input shape for testing the execution of the neural network model. The error code INCOMPATIBLE is returned if the execution fails. Setting aInputShape = 0 or aInputShape[0] = 0 disables the testing of the execution.
bRead	BOOL		Reading the file is triggered by a rising edge at this input.
nTimeout	TIME	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	BOOL	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	BOOL	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	UDINT	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Weiterführende Informationen

Der Funktionsblock `FB_VN_ReadNeuralNetwork` lädt ein neuronales Netz [▶ 445] aus einer Datei asynchron in die SPS. Die Datei muss im standardisierten Austauschformat Open Neural Network Exchange (ONNX) vorliegen, da nur dieses Format unterstützt wird.

Laden von geänderten neuronalen Netzen

Beim Einladen eines neuronalen Netzes cached der Vision-Service die Daten intern. Ein Einladen von geänderten neuronalen Netzen ist somit im Standard nicht möglich. Passen Sie dazu die Cache Einstellungen [▶ 68] an oder sehen Sie das Beispiel Serielles Laden von Bildern in die SPS [▶ 2922] für eine Alternative.

Parameter

Einige Parameter sind zentral im Kapitel File Access Parameter [▶ 1639] beschrieben.

Geladenes neuronales Netz (Rückgabewert)

Das geladene `ITcVnNeuralNetwork` ist nach erfolgreichem Lesevorgang und optionalem Test der Ausführung über den Interface Pointer `ipDestNeuralNetwork` zu erreichen.

Testen der Ausführung

Mit dem optionalen Parameter `aInputShape` kann ein Eingangsformat übergeben werden, mit dem die Ausführung des neuronalen Netzes, vor dem Laden in die SPS, getestet wird. Dieser Test ist hilfreich, insbesondere bei Modellen mit dynamischen Eingangsformaten, um sicher zu stellen, dass das Modell mit dem angegebenen Format erfolgreich ausgeführt werden kann. Falls die Ausführung nicht erfolgreich ist, wird ein `HRESULT = ADSERR_DEVICE_INCOMPATIBLE` zurückgegeben.

Mit der folgenden Angabe kann zum Beispiel ein Eingangsformat für den Test spezifiziert werden. Mit `aInputShape[0] := 0;` kann der Test der Ausführung ebenfalls deaktiviert werden.

```
aInputShape[0] := 4;
aInputShape[1] := 1;
aInputShape[2] := 224;
aInputShape[3] := 224;
aInputShape[4] := 3;
```

Anwendung

Das Laden eines neuronalen Netzes, ohne Test der Ausführung (`aInputShape := 0;`), vom Pfad `C:\TcVision\NeuralNetwork.onnx` sieht z. B. so aus:

```
fbReadNeuralNetwork(
    sFilePath           := 'C:\TcVision\NeuralNetwork.onnx',
    ipDestNeuralNetwork := ipReadNeuralNetwork,
    aInputShape         := 0;
    bRead               := TRUE,
    nTimeout            := T#500MS);

IF NOT fbReadNeuralNetwork.bBusy AND NOT fbReadNeuralNetwork.bError THEN
    // Neural Network is accessible in ipReadNeuralNetwork
END_IF
```

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.4 Image Acquisition

Diese Gruppe enthält Funktionsblöcke zur Bildaufnahme mit Vision-Geräten (GigE Vision Kamera [▶ 72] und File Source [▶ 135] Objekte).

Funktionsblöcke

- FB_VN_SimpleCameraControl [▶ 1697] für Kameras und File Sources
- FB_VN_GevCameraControl [▶ 1672] für GigE Vision Kameras
- FB_VN_FileSourceControl [▶ 1664] für File Sources

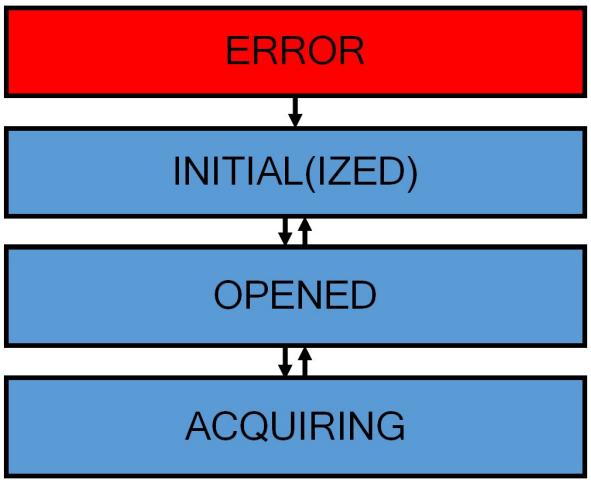
Allgemeines

Die Funktionsblöcke in dieser Gruppe stellen einen Proxy für die Image Provider TcCOM-Objekte dar. Dementsprechend müssen Sie nicht zyklisch aufgerufen werden. Stattdessen wird das TcCOM-Objekt über die Methoden des Funktionsblocks gesteuert.

Zustandsautomat

Alle Funktionsblöcke zur Bildaufnahme beinhalten einen Zustandsautomaten, um den Zustand des repräsentierten Geräts zu verwalten. Im Wesentlichen muss ein Funktionsblock in den ACQUIRING Zustand gebracht werden, um Bilder kontinuierlich zu empfangen. Der INITIAL Zustand stellt den Ausgangspunkt

nach einem Neustart oder Reset dar, von dem aus das INITIALIZING angestoßen werden kann. Ab dem OPENED-Zustand ist der Kommunikationskanal zu dem Gerät geöffnet, um z.B. Registerwerte auslesen oder schreiben zu können.



Alle Zustände werden durch das Enum ETcVnCameraState [▶ 179] repräsentiert und sind folgend erklärt:

Hauptzustände

Die Hauptzustände sind stationäre Zustände. Sie stellen im Wesentlichen dar, ob Verbindung zu einem Vision-Geräte besteht (`OPENED`) und ob das Gerät Bilder aufnimmt (`ACQUIRING`) sowie ob es einen Fehler bei der Verbindung mit dem Gerät gibt (`ERROR`). Zwischen diesen Zuständen können Sie mit den Methoden der FBs wechseln. Im Regelfall erfolgt der Übergang zwischen zwei Hauptzuständen nicht sofort, sondern über mindestens einen Zyklus. In dieser Zwischenzeit ist das Vision-Gerät in einem Zwischenzustand [▶ 1663].

Zustand	Beschreibung
ERROR	Fehlerzustand, aus dem jedes Gerät nur durch Aufruf der Methode `Reset` reaktiviert werden kann.
INITIAL	Initialer Zustand jedes Geräts.
INITIALIZED	Ähnlich wie INITIAL, wobei die Kamera manuell initialisiert wurde (z.B. mit Force-IP und den Initialization Commands)
OPENED	Es besteht eine Verbindung zur Kamera, einzelne Bilder können getriggert werden und Kameraregister können gelesen und beschrieben werden. **Dieser Zustand kann mit der Methode StartAcquisition übersprungen werden.**
ACQUIRING	Gerät befindet sich im Aufnahmezustand und sendet je nach Einstellung des Geräts Bilder.

Zwischenzustände

Die Zwischenzustände sind nötig, da die meisten Aktionen nicht sofort fertiggestellt werden können (z. B. die Bildaufnahme). Wenn sich die Funktionsblöcke in einem dieser Zwischenzustände befinden, muss die entsprechende Methode erneut aufgerufen werden, um den Übergang in den jeweiligen Hauptzustand zu vervollständigen.

Zustand	Beschreibung
INITIALIZING	INITIAL > INITIALIZED
OPENING	INITIAL > OPENED und INITIALIZED > OPENED
STARTACQUISITION	OPENED > ACQUIRING.
STOPACQUISITION	ACQUIRING > OPENED
RESETTINGFEATURES	OPENED > OPENED
TRIGGERING	OPENED > OPENED und ACQUIRING > ACQUIRING
CLOSING	OPENED > INITIALIZED

Methoden müssen in Zwischenzuständen aufgerufen werden!

Eine jeweilige Methode für Zustandsübergänge muss auch in den Zwischenzuständen aufgerufen werden. Erst wenn nach einem entsprechenden Methodenaufruf ein stationärer Zustand mit der Methode `GetState` festgestellt wird, ist der Zustandsübergang vollständig abgeschlossen und die Methode muss nicht mehr aufgerufen werden. Ein einmaliges Antriggern einer Methode und darauffolgendes Warten führt nicht zu dem gewünschten Zustandsübergang.

Ein Zustandsautomat in der SPS kann dann entweder mit einer `CASE … OF` Struktur oder mit einer `IF … ELSIF …` -Struktur implementiert werden.

Beispiele

- Bild per Dateiname triggern [▶ 2923]

6.1.5.4.1 FB_VN_FileSourceControl

This FB provides access to the images send by a File Source instance.

Do not call the main FB directly. Only use the available methods.

Methods

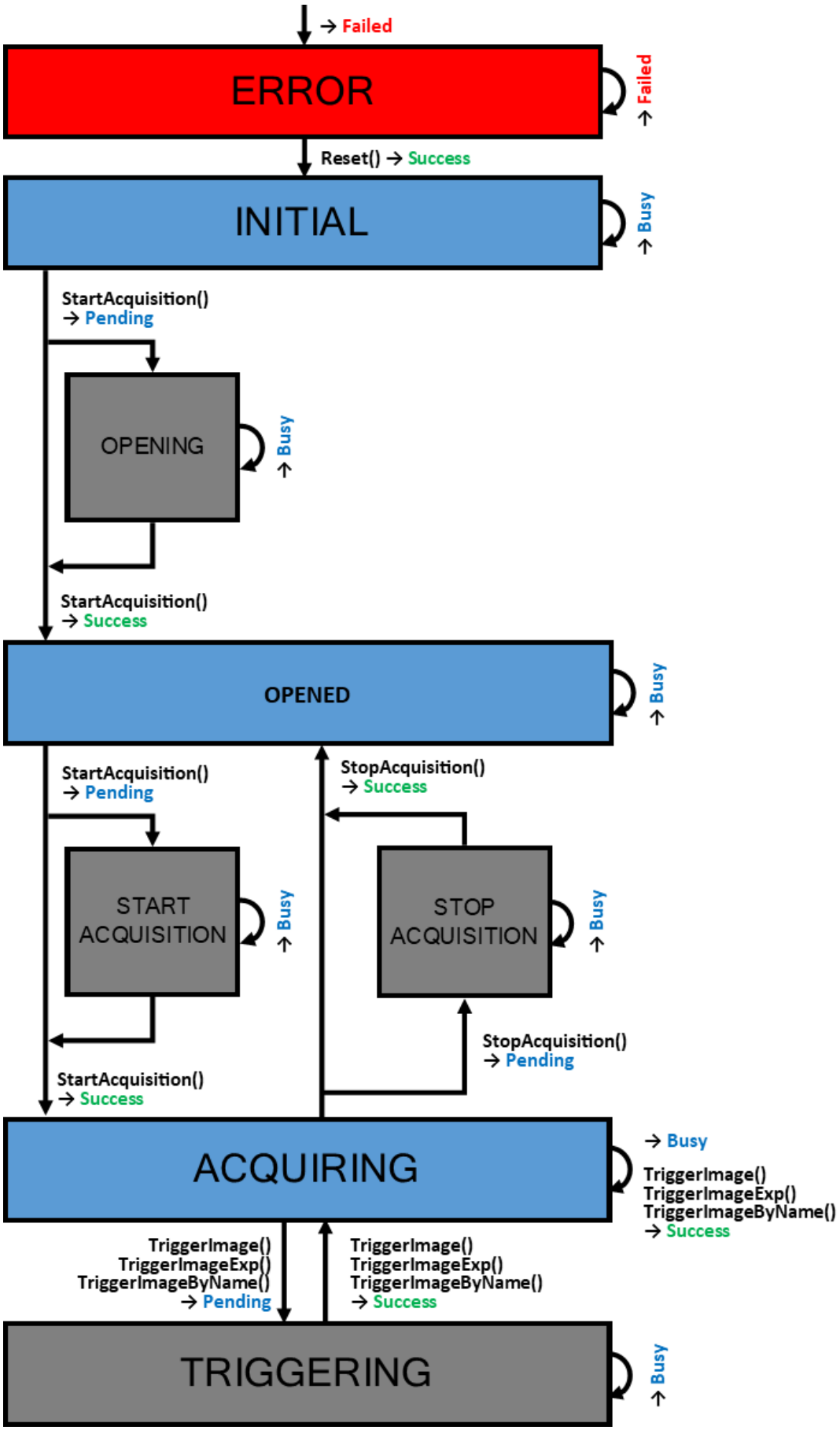
Name	Description
GetCurrentImage [▶ 1667]	Gets the current available image (if any).
GetCurrentImageAndFileName [▶ 1668]	Gets the current available image (if any) and the corresponding file name.
StartAcquisition [▶ 1668]	Activates the File Source. If not in trigger mode, it will start sending images.
StopAcquisition [▶ 1669]	Deactivates the File Source. It will stop sending images.
Reset [▶ 1670]	Reset the controller to initial state (might require multiple calls depending on current state, until S_OK is returned)
GetState [▶ 1670]	Gets the current state of the internal camera control state machine.
TriggerImage [▶ 1671]	If in trigger mode, this function triggers the next image in the list.
TriggerImageExp [▶ 1671]	If in trigger mode, this function triggers a specific image in the list.
TriggerImageByName [▶ 1672]	If in trigger mode, this function triggers a specific image in the list by name.

Weiterführende Informationen

Der Funktionsblock `FB_VN_FileSourceControl` kann File Source [▶ 135] Instanzen mit vollem Funktionsumfang in der SPS ansteuern. Alternativ ist dies mit beschränkter Funktionalität auch mit dem Funktionsbaustein FB_VN_SimpleCameraControl [▶ 1697] möglich.

Zustandsautomat

Die nachfolgende Grafik zeigt den Zustandsautomaten. Die Hauptzustände sind dabei blau gekennzeichnet, die Übergangszustände grau und der Error-Zustand rot.



Zur Steuerung des Zustandsautomats können folgende Methoden genutzt werden:

- GetState [▶ 1670]

- StartAcquisition [▶ 1668]
- StopAcquisition [▶ 1669]
- TriggerImage [▶ 1671]
- TriggerImageByName [▶ 1672]
- Reset [▶ 1670]

Wenn eine Methode in einem Zustand aufgerufen wird, für den sie nicht gedacht ist, wird der Fehlercode `INVALIDSTATE` (`16#712`) zurückgegeben.

Anwendung mit kontinuierlichem Bildeinzug

Ein Zustandsautomat, der kontinuierlich empfangene Bilder abruft, sieht z.B. so aus:

```
VAR
    fbFileSource     :   FB_VN_FileSourceControl;
    eState           :   ETcVnCameraState;
    ipImageIn        :   ITcVnImage;
    ipImageInDisp    :   ITcVnDisplayableImage;
    hr               :   HRESULT;
    nNewImageCounter:    UINT;
END_VAR

eState := fbFileSource.GetState();

CASE eState OF

TCVN_CS_INITIAL, TCVN_CS_OPENING, TCVN_CS_OPENED, TCVN_CS_STARTACQUISITION:
    hr := fbFileSource.StartAcquisition();

TCVN_CS_ACQUIRING:
    hr := fbFileSource.GetCurrentImage(ipImageIn);

    // Check if new Image was received
    IF SUCCEEDED(hr) AND ipImageIn <> 0 THEN
        nNewImageCounter := nNewImageCounter + 1;

        // Place to call vision algorithms
        hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, hr);
    END_IF

TCVN_CS_ERROR:
    hr := fbFileSource.Reset();

END_CASE
```

Anwendung mit Trigger

Ein Zustandsautomat, der kontinuierlich ein Bild mit einem speziellen Dateinamen triggert, sieht z.B. so aus:

```
VAR
    fbFileSource     :   FB_VN_FileSourceControl;
    eState           :   ETcVnCameraState;
    ipImageIn        :   ITcVnImage;
    ipImageInDisp    :   ITcVnDisplayableImage;
    bTrigger         :   BOOL := TRUE;
    sFileName        :   STRING := 'FileName.bmp';
    hr               :   HRESULT;
    nNewImageCounter:    UINT;
END_VAR

eState := fbFileSource.GetState();

CASE eState OF

TCVN_CS_INITIAL, TCVN_CS_OPENING, TCVN_CS_OPENED, TCVN_CS_STARTACQUISITION:
    hr := fbFileSource.StartAcquisition();

TCVN_CS_TRIGGERING:
    hr := fbFileSource.TriggerImage();

TCVN_CS_ACQUIRING:

    IF bTrigger THEN
        hr := fbFileSource.TriggerImageByName(sFileName);
```

```
        IF SUCCEEDED(hr) THEN
            bTrigger := FALSE;
        END_IF
    ELSE
        hr := fbFileSource.GetCurrentImage(ipImageIn);

        // Check if new Image was received
        IF SUCCEEDED(hr) AND ipImageIn <> 0 THEN
            nNewImageCounter := nNewImageCounter + 1;

            // Trigger next image
            bTrigger := TRUE;

            // Place to call vision algorithms
            hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, hr);
        END_IF
    END_IF

TCVN_CS_ERROR:
    hr := fbFileSource.Reset();

END_CASE
```

Alternativ zu der Methode `TriggerImageByName(sFileName)` kann die `TriggerImage()` verwendet werden, um immer das nächste Bild aus der Liste des File Source Controls zu erhalten.

Beispiele

- Bild per Dateiname triggern [▶ 2923]

Verwandte Funktionsblöcke

- FB_VN_SimpleCameraControl [▶ 1697] für Kameras und File Sources
- FB_VN_GevCameraControl [▶ 1672] für GigE Vision Kameras
- FB_VN_FileSourceControl [▶ 1664] für File Sources

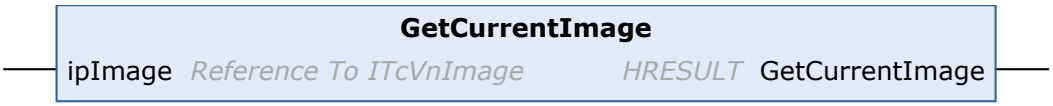
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.4.1.1 GetCurrentImage



Gets the current available image (if any).

Syntax

Definition:

```
METHOD GetCurrentImage : HRESULT
VAR_INPUT
    ipImage : Reference To ITcVnImage;
END_VAR
```

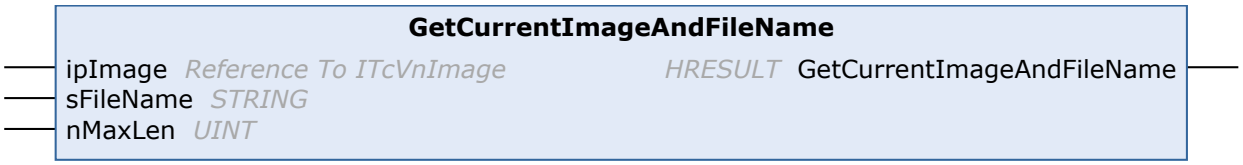
Inputs

Name	Type	Description
ipImage	Reference To ITcVnImage [▶ 426]	Returns the image

Return value

HRESULT [▶ 150]

6.1.5.4.1.2 GetCurrentImageAndFileName



Gets the current available image (if any) and the corresponding file name.

Syntax

Definition:

```
METHOD GetCurrentImageAndFileName : HRESULT
VAR_INPUT
    ipImage   : Reference To ITcVnImage;
    sFileName : STRING;
    nMaxLen   : UINT;
END_VAR
```

Inputs

Name	Type	Description
ipImage	Reference To ITcVnImage [▶ 426]	Returns the image
sFileName	STRING	Returns the file name
nMaxLen	UINT	Define the maximum allowed length for the file name

Return value

HRESULT [▶ 150]

6.1.5.4.1.3 StartAcquisition

Activates the File Source. If not in trigger mode, it will start sending images.

Syntax

Definition:

```
METHOD StartAcquisition : HRESULT
```

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Methode `StartAcquisition` startet das Einladen der Bilder. Abhängig von der Einstellung des Trigger-Modus kann dies bedeuten, direkt Bilder bereitzustellen oder auf ein Trigger-Signal zu warten. Das Starten des Bildeinladens kann aus den Zuständen `INITIAL` und `OPENED` geschehen.

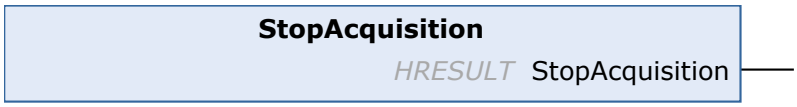
Zustandsübergänge

Mit `StartAcquisition` soll das File Source in den Zustand `ACQUIRING` versetzt werden. Dies kann aus den Zuständen `INITIAL` und `OPENED` heraus geschehen. Falls sich das File Source in `INITAL` befindet, wird zunächst der Zustandsübergang nach `OPENED` vollzogen. Dieser geschieht im Regelfall über den Zwischenzustand `OPENING`. Der Übergang von `OPENED` nach `ACQUIRING` geschieht im Regelfall über den Zwischenzustand `STARTACQUISITION`.

Alle möglichen Zustandsübergänge sowie ihre Return Codes sind in folgender Übergangsmatrix dargestellt:

Return Codes (`HRESULT`)		**Ergebniszustand**					
		INITIAL	**OPENING**	**OPENED**	**STARTACQUISITION**	**ACQUIRING**	**ERROR**
Ausgangszustand	**INITIAL**	`BUSY`	`PENDING`	`SUCCESS`	-	-	`FAILED`
	OPENING	-	`BUSY`				
	OPENED		-	`BUSY`	`PENDING`	`SUCCESS`	
	STARTACQUISITION			-	`BUSY`		

6.1.5.4.1.4 StopAcquisition



Deactivates the File Source. It will stop sending images.

Syntax

Definition:

```
METHOD StopAcquisition : HRESULT
```

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Methode `StopAcquisition` stoppt das Bildeinladen des File Source.

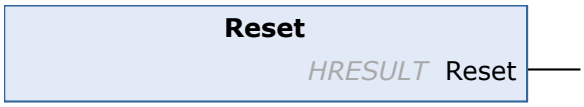
Zustandsübergänge

Mit `StopAcquisition` soll das File Source aus dem Zustand `ACQUIRING` in den Zustand `OPENED` gebracht werden. Im Regelfall geschieht dieser Übergang nicht sofort, sondern über den Zwischenzustand `STOPACQUISITION`.

Alle möglichen Zustandsübergänge sowie ihre Return Codes sind in folgender Übergangsmatrix dargestellt:

Return Codes (HRESULT)		Ergebniszustand			
		ACQUIRING	STOPACQUISITION	OPENED	ERROR
Ausgangszustand	ACQUIRING	BUSY	PENDING	SUCCESS	FAILED
	STOPACQUISITION	-	BUSY		

6.1.5.4.1.5 Reset



Reset the controller to initial state (might require multiple calls depending on current state, until S_OK is returned)

Syntax

Definition:

```
METHOD Reset : HRESULT
```

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Methode `Reset` setzt das File Source zurück, wenn es im Fehlerzustand ist.

Zustandsübergänge

Mit `Reset` soll das File Source vom Zustand `ERROR` in den Zustand `INITIAL` gebracht werden. Wenn das Rücksetzen erfolgreich ist, geschieht dieser Übergang sofort, ansonsten bleibt der Funktionsblock im Zustand `ERROR`.

Alle möglichen Zustandsübergänge sowie ihre Return Codes sind in folgender Übergangsmatrix dargestellt:

Return Codes (HRESULT)		Ergebniszustand	
		ERROR	INITIAL
Ausgangszustand	ERROR	FAILED	SUCCESS

Im Erfolgsfall wird S_OK oder S_FALSE (wenn kein Fehler zurückzusetzten war) als SUCCESS-Code zurückgegeben.

6.1.5.4.1.6 GetState

Gets the current state of the internal camera control state machine.

Syntax

Definition:

```
METHOD GetState : ETcVnCameraState
```

Return value

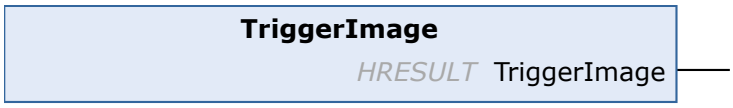
ETcVnCameraState [▶ 179]

Weiterführende Informationen

Die Methode `GetState` gibt den Zustand des Kameraobjekts bzw. File Source Objekts als Wert vom Enum ETcVnCameraState [▶ 179] zurück. Folgende Zustände sind möglich:

- ERROR
- INITIAL
- OPENING
- OPENED
- STARTACQUISITION
- STOPACQUISITION
- ACQUIRING
- TRIGGERING

6.1.5.4.1.7 TriggerImage



If in trigger mode, this function triggers the next image in the list.

Syntax

Definition:

```
METHOD TriggerImage : HRESULT
```

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Methode TriggerImage triggert ein Bild aus dem File Source.

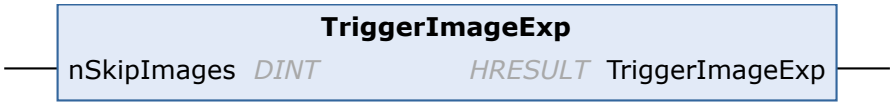
Zustandsübergänge

Mit TriggerImage soll das File Source nur ein Bild einladen und dabei im aktuellen Zustand ACQUIRING bleiben. Da die Bildaufnahme üblicherweise mehrere Zyklen dauert, geht das File Source im Regelfall jedoch kurzzeitig in den Zwischenzustand `TRIGGERING`.

Alle möglichen Zustandsübergänge sowie ihre Return Codes sind in folgender Übergangsmatrix dargestellt:

Return Codes (`HRESULT`)		**Ergebniszustand**		
		TRIGGERING	**ACQUIRING**	**ERROR**
Ausgangszustand	**ACQUIRING**	`PENDING`	`BUSY / SUCCESS`	`FAILED`
	TRIGGERING	`BUSY`	`SUCCESS`	

6.1.5.4.1.8 TriggerImageExp



If in trigger mode, this function triggers a specific image in the list.

Syntax

Definition:

```
METHOD TriggerImageExp : HRESULT
VAR_INPUT
    nSkipImages : DINT;
END_VAR
```

Inputs

Name	Type	Description
nSkipImages	DINT	Amount of images to skip, relative to current

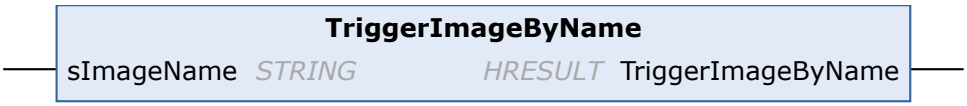
Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Zustandsübergänge der Methode `TriggerImageExp` sind die gleichen wie bei TriggerImage [▶ 1671].

6.1.5.4.1.9 TriggerImageByName



If in trigger mode, this function triggers a specific image in the list by name.

Syntax

Definition:

```
METHOD TriggerImageByName : HRESULT
VAR_INPUT
    sImageName : STRING;
END_VAR
```

Inputs

Name	Type	Description
sImageName	STRING	Name of the image to trigger

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Zustandsübergänge der Methode `TriggerImageByName` sind die gleichen wie bei TriggerImage [▶ 1671].

6.1.5.4.2 FB_VN_GevCameraControl

This FB provides the basic functionality to control a GigE Vision camera and access its calibration data.

Do not call the main FB directly. Only use the available methods.

Methods

Name	Description
ClearImageQueue [▶ 1679]	Delete all images contained in the TcVnGevImageProvider TcCOM module receive queue and reset the corresponding omitted images counter.
CloseCamera [▶ 1679]	Close the control channel to the camera.
GetCalibPatternRef [▶ 1680]	Gets the reference calibration pattern points from the TcVnGevImageProvider TcCOM module (Can be set from the calibration assistant)
GetCameraMatrix [▶ 1681]	Gets the camera matrix from the TcVnGevImageProvider TcCOM module (requires intrinsic calibration parameters, e.g. set by the calibration assistant)
GetCurrentImage [▶ 1681]	Gets the current available image (first in receive queue).
GetCurrentImageUndistorted [▶ 1681]	Gets the current available image (first in receive queue) with undistortion applied. Incompatible with pixel formats that contain an encoding, e.g. a Bayer pattern. Requires intrinsic calibration parameters in the TcVnGevImageProvider TcCOM module, e.g. set by the calibration assistant.
GetCurrentImageWithGvspInfo [▶ 1682]	Gets the current available image (first in receive queue) and its corresponding GVSP info.
GetCurrentImageWithGvspInfo Undistorted [▶ 1683]	Gets the current available image (first in receive queue) with its corresponding GVSP info and undistortion applied. Incompatible with pixel formats that contain an encoding, e.g. a Bayer pattern. Requires intrinsic calibration parameters in the TcVnGevImageProvider TcCOM module, e.g. set by the calibration assistant.
GetCurrentImageWithTimestamps [▶ 1683]	Gets the current available image (first in receive queue) and corresponding timestamps.
GetDistortionCoefficients [▶ 1684]	Gets the distortion coefficients from the TcVnGevImageProvider TcCOM module (requires intrinsic calibration parameters, e.g. set by the calibration assistant)
GetLastImageFromQueue [▶ 1684]	Gets the last received image from the queue.
GetOmittedImagesNum [▶ 1685]	Gets the number of omitted images since the last call of ClearImageQueue(). If the image receive queue in the TcVnGevImageProvider TcCOM module is full and a new image arrives, the first one in the queue will be deleted and the omitted counter is increased.
GetRotationMatrix [▶ 1685]	Gets the rotation matrix from the TcVnGevImageProvider TcCOM module (requires extrinsic calibration parameters, e.g. set by the calibration assistant)
GetState [▶ 1686]	Gets the current state of the internal camera control state machine.
GetTranslationVector [▶ 1686]	Gets the translation vector from the TcVnGevImageProvider TcCOM module (requires extrinsic calibration parameters, e.g. set by the calibration assistant)
InitializeCamera [▶ 1687]	Initialize the camera to the intended state (includes sending the 'InitCameraCommands' defined in the GevImageAcquisition TcCOM module to the camera).
OpenCamera [▶ 1688]	Open a control channel to the camera.
Reset [▶ 1689]	Reset the camera controller to initial state (might require multiple calls depending on current state, until S_OK is returned)
ResetCameraFeatures [▶ 1690]	Reset the camera features to initial state (sends the 'InitCameraCommands' defined in the GevImageAcquisition TcCOM module to the camera).
SetCameraMatrix [▶ 1691]	Sets the camera matrix to the TcVnGevImageProvider TcCOM module
SetDistortionCoefficients [▶ 1692]	Sets the distortion coefficients to the TcVnGevImageProvider TcCOM module

Name	Description
SetRotationMatrix [▶ 1692]	Sets the rotation matrix to the TcVnGevImageProvider TcCOM module
SetTranslationVector [▶ 1692]	Sets the translation vector to the TcVnGevImageProvider TcCOM module
StartAcquisition [▶ 1693]	Send the 'StartAcquisitionCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.
StopAcquisition [▶ 1695]	Send the 'StopAcquisitionCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.
TriggerImage [▶ 1696]	Send the 'SoftwareTriggerCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.

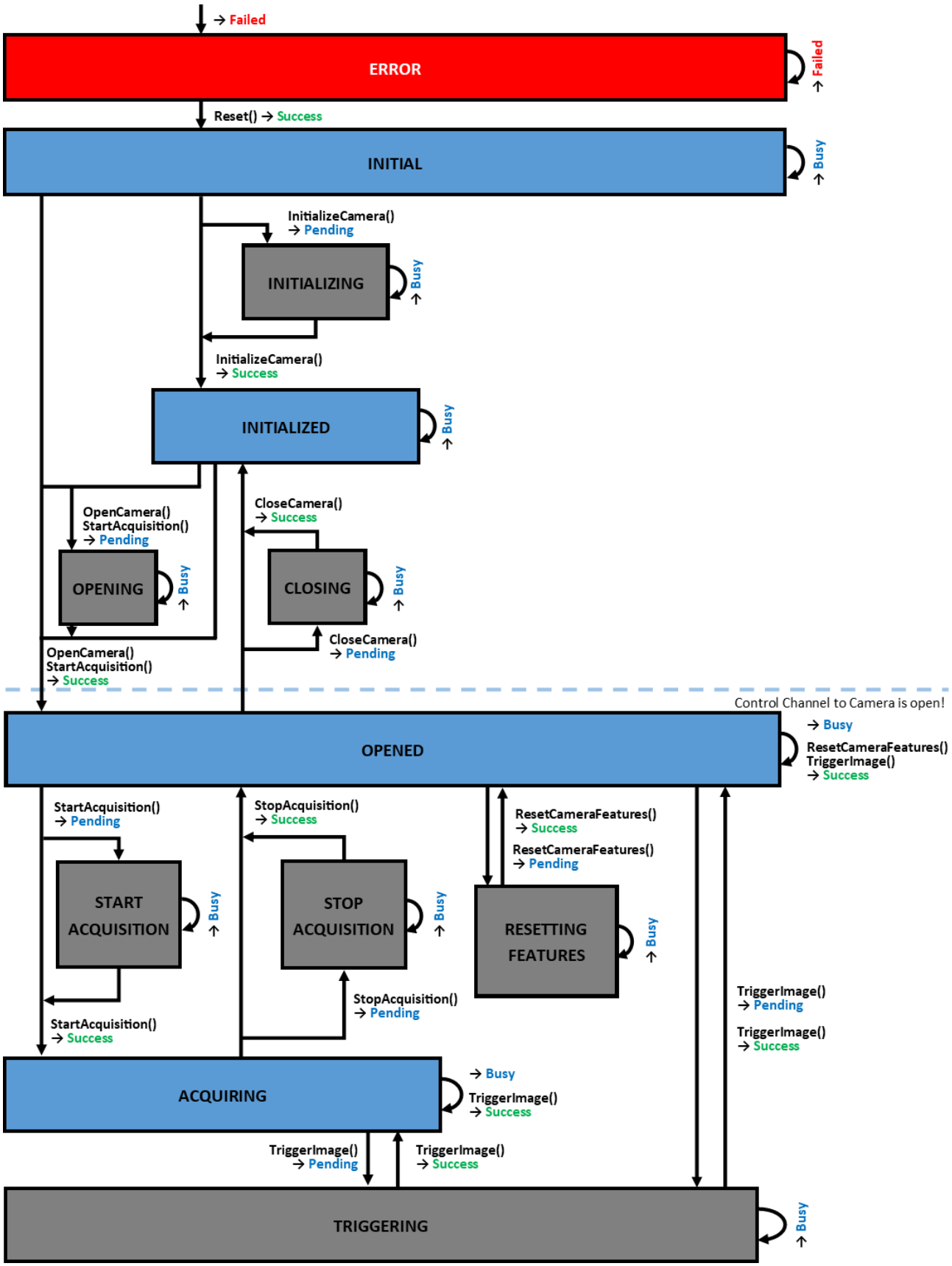
Weiterführende Informationen

Der Funktionsblock `FB_VN_GevCameraControl` repräsentiert GigE Vision Camera [▶ 72] Objekte im vollen Umfang in der SPS. Alternativ kann für einen beschränkten Zugriff auf Kamera Objekte auch der Funktionsblock FB_VN_SimpleCameraControl [▶ 1697] genutzt werden.

Dieser Funktionsblock ist nicht dafür gedacht, direkt aufgerufen zu werden. Vielmehr enthalten die Methoden sämtliche Funktionalität.

Zustandsautomat

Die nachfolgende Grafik zeigt den Zustandsautomaten. Die Hauptzustände sind dabei blau gekennzeichnet, die Übergangszustände grau und der Error-Zustand rot. In welchem Zustand sich die Kamera befindet, kann über die Methode GetState [▶ 1686] abgefragt werden. Zudem ist dargestellt, mit welchen Methoden ein Zustandsübergang erreicht werden kann und zu welchem Rückgabewert dieser jeweils führt. Wenn Init Commands vorhanden sind und der InitializationAutoMode AUTOINIT_AFTER_SO ausgewählt wurde, wird auch kurzzeitig der State INITIALIZING angenommen, in dem die automatische Initialisierung erfolgt.



Den aktuellen Zustand können Sie mit der Methode GetState abfragen.

Zur Steuerung des Zustandsautomaten können folgende Methoden verwendet werden:

- GetState [▶ 1686]
- InitializeCamera [▶ 1687]

- OpenCamera [▶ 1688]
- CloseCamera [▶ 1679]
- StartAcquisition [▶ 1693]
- StopAcquisition [▶ 1695]
- TriggerImage [▶ 1696]
- ResetCameraFeatures [▶ 1690]
- Reset [▶ 1689]

Wenn eine Methode in einem Zustand aufgerufen wird, für den sie nicht gedacht ist, wird der Fehlercode `INVALIDSTATE` (`16#712`) zurückgegeben.

HRESULT

Hex	Dec	Code	Beschreibung
16#718	1816	NOTINIT	Kamera wurde noch nicht initialisiert. Initialisieren Sie die Kamera entweder mit der Methode InitializeCamera [▶ 1687] oder indem Sie den TcCOM-Parameter InitializationAutoMode [▶ 106] auf AUTOINIT_SO oder AUTOINIT_AFTER_SO setzen.
16#71A	1818	NOINTERFACE	Der Kamera-Funktionsblock ist nicht mit dem Kamera-TcCOM-Objekt verknüpft. Stellen Sie diese Verknüpfung in der TwinCAT Projektinstanz unter *Symbol Initialization* her.

Anwendung mit kontinuierlichem Bildeinzug

Ein Zustandsautomat, um eine Kamera in den Bildaufnahme-Modus zu bringen, Fehlerzustände zu behandeln und kontinuierlich erfolgreich aufgenommene Bilder anzuzeigen, sieht z. B. so aus:

```
VAR
    hr              :   HRESULT;
    ipImageIn       :   ITcVnImage;
    ipImageInDisp   :   ITcVnDisplayableImage;
    fbCameraControl :   FB_VN_GevCameraControl;
    eCameraState    :   ETcVnCameraState;
    nNewImageCounter:   UINT;
END_VAR

eCameraState :=  fbCameraControl.GetState();

// CameraControl is in error state, so try to reset the camera connection
IF eCameraState = TCVN_CS_ERROR THEN
    hr :=  fbCameraControl.Reset();

// Camera not yet initialized
ELSIF eCameraState < TCVN_CS_INITIALIZED THEN
    hr :=  fbCameraControl.InitializeCamera();

// Camera not yet opened
ELSIF eCameraState < TCVN_CS_OPENED THEN
    hr := fbCameraControl.OpenCamera();

// Camera not yet streaming
ELSIF eCameraState < TCVN_CS_ACQUIRING THEN
    hr := fbCameraControl.StartAcquisition();

// Camera streaming
ELSIF eCameraState = TCVN_CS_ACQUIRING THEN
    hr := fbCameraControl.GetCurrentImage(ipImageIn);

    // Check if new Image was received
    IF SUCCEEDED(hr) AND ipImageIn <> 0 THEN
        nNewImageCounter := nNewImageCounter + 1;

        // Place to call vision algorithms
        hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, hr);
    END_IF

END_IF
```

In diesem Fall kann die Kamera auf kontinuierliche Bild-Akquise [▶ 2938] oder auf einen Hardwaretrigger [▶ 2938] konfiguriert werden.

Anwendung mit Softwaretrigger

Ein Zustandsautomat, um eine Kamera in den Bildaufnahme-Modus zu bringen, Fehlerzustände zu behandeln und per Softwaretrigger eine Bildaufnahme anzutriggern, sieht z. B. so aus:

```
VAR
    hr              :   HRESULT;
    ipImageIn       :   ITcVnImage;
    ipImageInDisp   :   ITcVnDisplayableImage;
    fbCameraControl :   FB_VN_GevCameraControl;
    eCameraState    :   ETcVnCameraState;
    nNewImageCounter:   UINT;
    bTrigger        :   BOOL := TRUE;
END_VAR

eCameraState :=  fbCameraControl.GetState();

// CameraControl is in error state, so try to reset the camera connection
IF eCameraState = TCVN_CS_ERROR THEN
    hr :=  fbCameraControl.Reset();

// Camera trigger image
ELSIF eCameraState = TCVN_CS_TRIGGERING THEN
    hr := fbCameraControl.TriggerImage();

// Camera not yet initialized
ELSIF eCameraState < TCVN_CS_INITIALIZED THEN
    hr :=  fbCameraControl.InitializeCamera();

// Camera not yet opened
ELSIF eCameraState < TCVN_CS_OPENED THEN
    hr := fbCameraControl.OpenCamera();

// Camera not yet streaming
ELSIF eCameraState < TCVN_CS_ACQUIRING THEN
    hr := fbCameraControl.StartAcquisition();

// Camera streaming
ELSIF eCameraState = TCVN_CS_ACQUIRING THEN

    IF bTrigger THEN
        hr := fbCameraControl.TriggerImage();
        IF SUCCEEDED(hr) THEN
            bTrigger := FALSE;
        END_IF
    ELSE
        hr := fbCameraControl.GetCurrentImage(ipImageIn);

        // Check if new Image was received
        IF SUCCEEDED(hr) AND ipImageIn <> 0 THEN
            nNewImageCounter := nNewImageCounter + 1;

            // Trigger next image
            bTrigger := TRUE;

            // Place to call vision algorithms
            hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, hr);
        END_IF
    END_IF

END_IF
```

In diesem Fall muss die Kamera auf einen Softwaretrigger [▶ 2938] (TriggerSource = Software) konfiguriert werden.

Verwandte Funktionsblöcke

- FB_VN_SimpleCameraControl [▶ 1697] für Kameras und File Sources
- FB_VN_GevCameraControl [▶ 1672] für GigE Vision Kameras
- FB_VN_FileSourceControl [▶ 1664] für File Sources

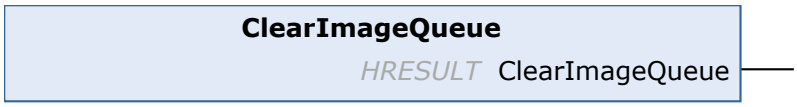
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.4.2.1 ClearImageQueue



Delete all images contained in the TcVnGevImageProvider TcCOM module receive queue and reset the corresponding omitted images counter.

Syntax

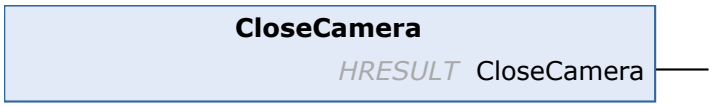
Definition:

```
METHOD ClearImageQueue : HRESULT
```

Return value

HRESULT [▶ 150]

6.1.5.4.2.2 CloseCamera



Close the control channel to the camera.

Syntax

Definition:

```
METHOD CloseCamera : HRESULT
```

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Methode `CloseCamera` schließt einen offenen Kommunikationskanal zu einer Kamera.

Zustandsübergänge

Mit `CloseCamera` soll die Kamera aus dem Zustand `OPENED` in den Zustand `INITIALIZED` gebracht werden. Im Regelfall geschieht dieser Übergang über den Zwischenzustand `CLOSING`.

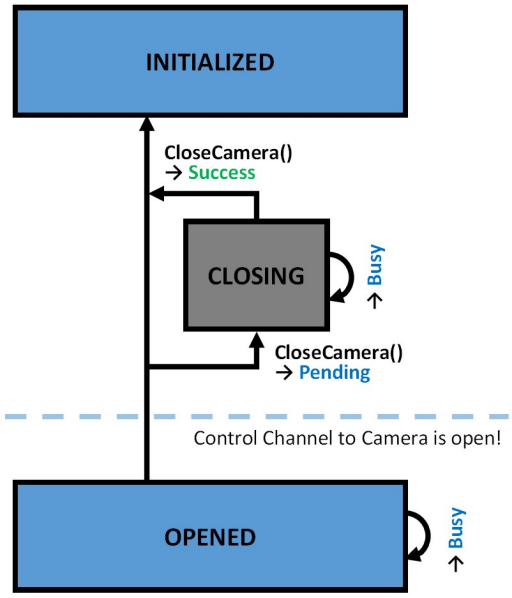


Abb. 19: Zustandsübergänge bei FB_VN_GevCameraControl.CloseCamera

Alle möglichen Zustandsübergänge sowie ihre Return Codes sind in folgender Übergangsmatrix dargestellt:

Return Codes (HRESULT)		**Ergebniszustand**			
		OPENED	**CLOSING**	**INITIALIZED**	**ERROR**
Ausgangszustand	**OPENED**	BUSY	PENDING	SUCCESS	FAILED
	CLOSING	-	BUSY		

6.1.5.4.2.3 GetCalibPatternRef

Gets the reference calibration pattern points from the TcVnGevImageProvider TcCOM module (Can be set from the calibration assistant)

Syntax

Definition:

```
METHOD GetCalibPatternRef : HRESULT
VAR_INPUT
    ipCalibPatternRef : Reference To ITcVnContainer;
END_VAR
```

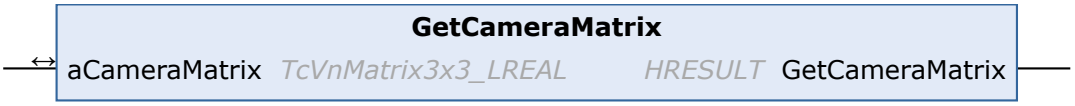
Inputs

Name	Type	Description
ipCalibPatternRef	Reference To ITcVnContainer [▶ 385]	Returns the reference calibration pattern points (ContainerType_Vector_TcVnPoint3_REAL)

Return value

HRESULT [▶ 150]

6.1.5.4.2.4 GetCameraMatrix



Gets the camera matrix from the TcVnGevImageProvider TcCOM module (requires intrinsic calibration parameters, e.g. set by the calibration assistant)

Syntax

Definition:

```
METHOD GetCameraMatrix : HRESULT
VAR_IN_OUT
    aCameraMatrix : TcVnMatrix3x3_LREAL;
END_VAR
```

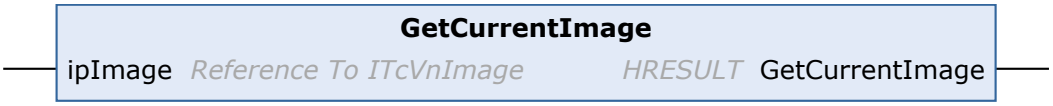
In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns the camera matrix

Return value

HRESULT [▶ 150]

6.1.5.4.2.5 GetCurrentImage



Gets the current available image (first in receive queue).

Syntax

Definition:

```
METHOD GetCurrentImage : HRESULT
VAR_INPUT
    ipImage : Reference To ITcVnImage;
END_VAR
```

Inputs

Name	Type	Description
ipImage	Reference To ITcVnImage [▶ 426]	Returns the image

Return value

HRESULT [▶ 150]

6.1.5.4.2.6 GetCurrentImageUndistorted

Gets the current available image (first in receive queue) with undistortion applied. Incompatible with pixel formats that contain an encoding, e.g. a Bayer pattern. Requires intrinsic calibration parameters in the TcVnGevImageProvider TcCOM module, e.g. set by the calibration assistant.

Syntax

Definition:

```
METHOD GetCurrentImageUndistorted : HRESULT
VAR_INPUT
    ipImage : Reference To ITcVnImage;
END_VAR
```

Inputs

Name	Type	Description
ipImage	Reference To ITcVnImage [▶ 426]	Returns the undistorted image

Return value

HRESULT [▶ 150]

6.1.5.4.2.7 GetCurrentImageWithGvspInfo

Gets the current available image (first in receive queue) and its corresponding GVSP info.

Syntax

Definition:

```
METHOD GetCurrentImageWithGvspInfo : HRESULT
VAR_INPUT
    ipImage    : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    stGvspInfo : GVSP_IMAGE_INFO;
END_VAR
```

Inputs

Name	Type	Description
ipImage	Reference To ITcVnImage [▶ 426]	Returns the image

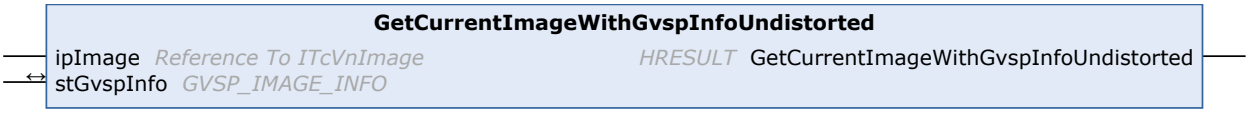
In/Outputs

Name	Type	Description
stGvspInfo	GVSP_IMAGE_INFO [▶ 238]	Returns the GVSP info

Return value

HRESULT [▶ 150]

6.1.5.4.2.8 GetCurrentImageWithGvspInfoUndistorted



Gets the current available image (first in receive queue) with its corresponding GVSP info and undistortion applied. Incompatible with pixel formats that contain an encoding, e.g. a Bayer pattern. Requires intrinsic calibration parameters in the TcVnGevImageProvider TcCOM module, e.g. set by the calibration assistant.

Syntax

Definition:

```
METHOD GetCurrentImageWithGvspInfoUndistorted : HRESULT
VAR_INPUT
    ipImage    : Reference To ITcVnImage;
END_VAR
VAR_IN_OUT
    stGvspInfo : GVSP_IMAGE_INFO;
END_VAR
```

Inputs

Name	Type	Description
ipImage	Reference To ITcVnImage [▶ 426]	Returns the undistorted image

In/Outputs

Name	Type	Description
stGvspInfo	GVSP_IMAGE_INFO [▶ 238]	Returns the GVSP info

Return value

HRESULT [▶ 150]

6.1.5.4.2.9 GetCurrentImageWithTimestamps

Gets the current available image (first in receive queue) and corresponding timestamps.

Syntax

Definition:

```
METHOD GetCurrentImageWithTimestamps : HRESULT
VAR_INPUT
    ipImage     : Reference To ITcVnImage;
    nTimestamps : ULINT;
    pTimestamps : Pointer To LINT;
END_VAR
VAR_OUTPUT
    stGvspInfo  : GVSP_IMAGE_INFO;
END_VAR
```

Inputs

Name	Type	Description
ipImage	Reference To ITcVnImage [▶ 426]	Returns the image
nTimestamps	ULINT	Select timestamps to return (ETcVnTimestamp)
pTimestamps	Pointer To LINT	Returns the requested timestamps (make sure to provide an array of sufficient size).

Outputs

Name	Type	Description
stGvspInfo	GVSP_IMAGE_INFO [▶ 238]	Returns the GVSP info

Return value

HRESULT [▶ 150]

6.1.5.4.2.10 GetDistortionCoefficients

Gets the distortion coefficients from the TcVnGevImageProvider TcCOM module (requires intrinsic calibration parameters, e.g. set by the calibration assistant)

Syntax

Definition:

```
METHOD GetDistortionCoefficients : HRESULT
VAR_IN_OUT
    aDistortionCoefficients : TcVnArray8_LREAL;
END_VAR
```

In/Outputs

Name	Type	Description
aDistortionCoefficients	TcVnArray8_LREAL [▶ 173]	Returns the distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]

Return value

HRESULT [▶ 150]

6.1.5.4.2.11 GetLastImageFromQueue

Gets the last received image from the queue.

Syntax

Definition:

```
METHOD GetLastImageFromQueue : HRESULT
VAR_INPUT
    ipImage : Reference To ITcVnImage;
END_VAR
```

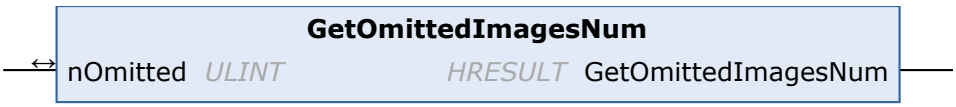
Inputs

Name	Type	Description
ipImage	Reference To ITcVnImage [▶ 426]	Returns the image

Return value

HRESULT [▶ 150]

6.1.5.4.2.12 GetOmittedImagesNum



Gets the number of omitted images since the last call of ClearImageQueue(). If the image receive queue in the TcVnGevImageProvider TcCOM module is full and a new image arrives, the first one in the queue will be deleted and the omitted counter is increased.

Syntax

Definition:

```
METHOD GetOmittedImagesNum : HRESULT
VAR_IN_OUT
    nOmitted : ULINT;
END_VAR
```

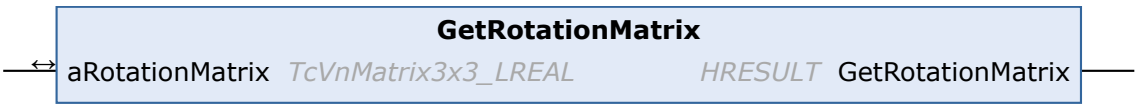
In/Outputs

Name	Type	Description
nOmitted	ULINT	Returns the number of omitted images

Return value

HRESULT [▶ 150]

6.1.5.4.2.13 GetRotationMatrix



Gets the rotation matrix from the TcVnGevImageProvider TcCOM module (requires extrinsic calibration parameters, e.g. set by the calibration assistant)

Syntax

Definition:

```
METHOD GetRotationMatrix : HRESULT
VAR_IN_OUT
    aRotationMatrix : TcVnMatrix3x3_LREAL;
END_VAR
```

In/Outputs

Name	Type	Description
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	Returns the rotation matrix

Return value

HRESULT [▶ 150]

6.1.5.4.2.14 GetState

Gets the current state of the internal camera control state machine.

Syntax

Definition:

```
METHOD GetState : ETcVnCameraState
```

Return value

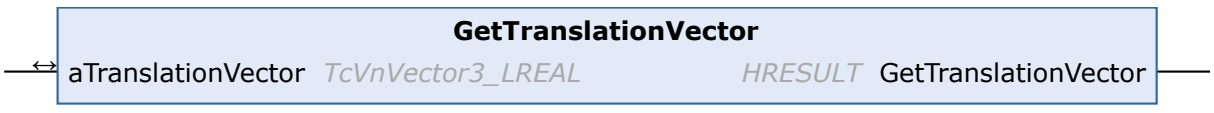
ETcVnCameraState [▶ 179]

Weiterführende Informationen

Die Methode `GetState` gibt den Zustand des Kameraobjekts als Wert vom Enum ETcVnCameraState [▶ 179] zurück. Folgende Zustände sind möglich:

- ERROR
- INITIAL
- INITIALIZING
- INITIALIZED
- OPENING
- OPENED
- CLOSING
- STARTACQUISITION
- STOPACQUISITION
- ACQUIRING
- TRIGGERING
- RESETTINGFEATURES

6.1.5.4.2.15 GetTranslationVector



Gets the translation vector from the TcVnGevImageProvider TcCOM module (requires extrinsic calibration parameters, e.g. set by the calibration assistant)

Syntax

Definition:

```
METHOD GetTranslationVector : HRESULT
VAR_IN_OUT
    aTranslationVector : TcVnVector3_LREAL;
END_VAR
```

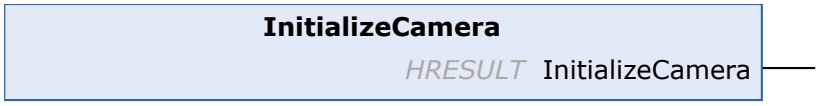
In/Outputs

Name	Type	Description
aTranslationVector	TcVnVector3_LREAL [▶ 173]	Returns the translation vector

Return value

HRESULT [▶ 150]

6.1.5.4.2.16 InitializeCamera



Initialize the camera to the intended state (includes sending the 'InitCameraCommands' defined in the GevImageAcquisition TcCOM module to the camera).

Syntax

Definition:

```
METHOD InitializeCamera : HRESULT
```

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Methode `InitializeCamera` initialisiert eine verbundene GigE Vision Kamera, so wie die Parameter in den hinterlegten Initialization Commands konfiguriert wurden. Diese Methode sollte benutzt werden, wenn explizit gesteuert werden soll, wann die Kamera initialisiert wird. Dazu muss der `InitializationAutoMode` in den TcCOM-Parametern des Image Acquisition Objekts [▶ 107] auf `NO_AUTOINIT` gesetzt werden. Dies bewirkt, dass keine automatische Kamera-Initialisierung ausgeführt wird, sondern diese manuell mittel `InitializeCamera` vorgenommen werden muss.

Falls eine andere Auswahl bei `InitializationAutoMode` getroffen wurde, erfolgt eine automatische Initialisierung und die Methode `InitializeCamera` wird nicht benötigt

Die Initialisierung eine GigE Vision Kamera besteht aus den folgenden Schritten:

1. Öffnen des Control Channels zur Kamera
2. Ausführen den Force-IP Kommandos (optional)
3. Setzen eines UserSets oder Schreiben von Initialization Commands [▶ 94].

Während des Initialisierens nimmt der Funktionsblock den Zustand INITIALIZING an. Sobald die Initialisierung abgeschlossen ist, geht der Funktionsblock in den Zustand INITIALIZED. Diesen nimmt er im Übrigen auch an, wenn die Kamera-Verbindung im Zustand OPENED durch Aufruf der Methode CloseCamera [▶ 1679] geschlossen wird.

Zustandsübergänge

Mit `InitializeCamera` soll die Kamera von dem Zustand `INITIAL` in den Zustand `INITIALIZED` gebracht werden. Im Regelfall geschieht dieser Übergang über den Zwischenzustand `INITIALIZING`.

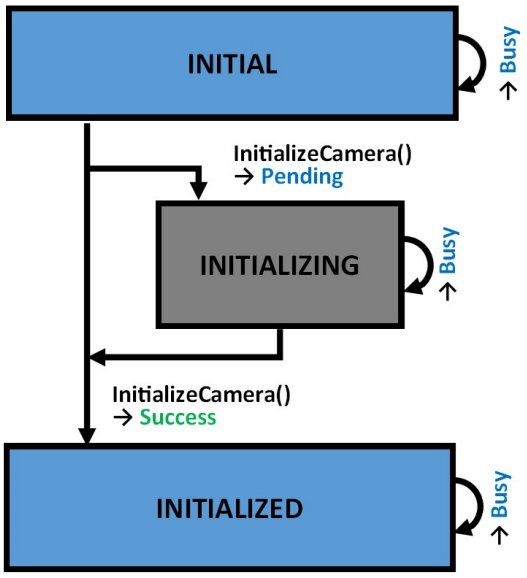
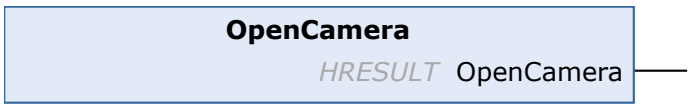


Abb. 20: Zustandsübergänge mit FB_VN_GevCameraControl.InitializeCamera

Alle möglichen Zustandsübergänge sowie ihre Return Codes sind in folgender Übergangsmatrix dargestellt:

Return Codes (`HRESULT`)		**Ergebniszustand**			
		INITIAL	**INITIALIZING**	**INITIALIZED**	**ERROR**
Ausgangszustand	**INITIAL**	`BUSY`	`PENDING`	`SUCCESS`	`FAILED`
	INITIALIZING	-	`BUSY`		

6.1.5.4.2.17 OpenCamera



Open a control channel to the camera.

Syntax

Definition:

```
METHOD OpenCamera : HRESULT
```

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Methode `OpenCamera` öffnet einen Kommunikationskanal zur Kamera. In dem entsprechenden Zustand `OPENED` können Kameraregister gelesen und geschrieben werden, Bilder aufgenommen werden sowie die kontinuierliche Bildaufnahme gestartet werden.

Falls der `InitializationAutoMode` in den TcCOM-Parametern des Image Acquisition Objekts [▶ 107] auf `AUTOINIT_AFTER_SO` steht wird beim ersten Aufruf der Methode OpenCamera die GigE Vision Kamera initialisiert. Dies beinhaltet u. a. das Schreiben der Initialization Commands [▶ 94] oder das Setzen der UserSets. Alternativ kann die Methode InitializeCamera [▶ 1687] aufgerufen werden, um die Initialisierung manuell vorzunehmen.

Zustandsübergänge

Mit `OpenCamera` soll die Kamera in den Zustand `OPENED` gebracht werden. Dies kann aus den Zuständen `INITIAL` und `INITIALIZED` heraus geschehen. Im Regelfall geschieht der Übergang über den Zwischenzustand `OPENING`.

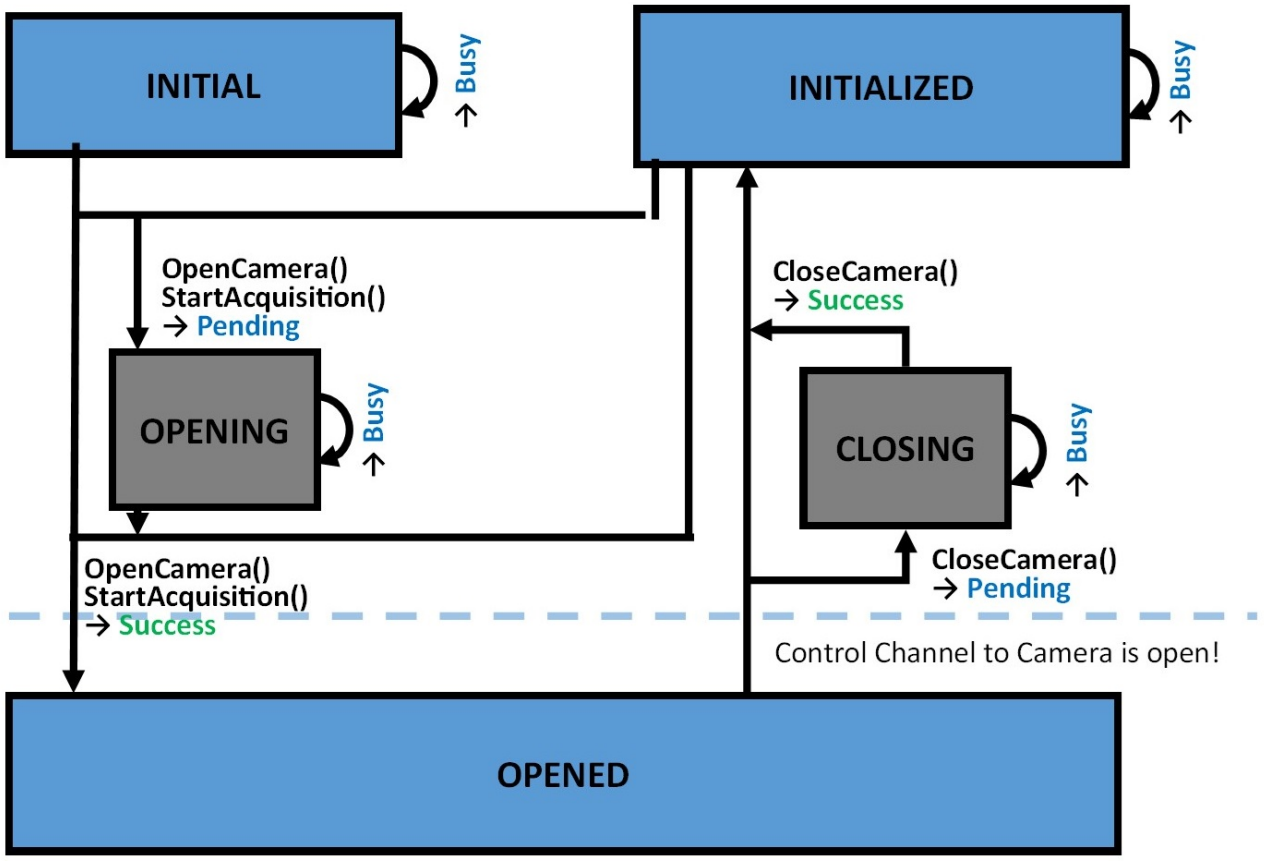


Abb. 21: Zustandsübergänge mit FB_VN_GevCameraControl.OpenCamera

Alle möglichen Zustandsübergänge sowie ihre Return Codes sind in folgender Übergangsmatrix dargestellt:

Return Codes (`HRESULT`)		**Ergebniszustand**				
		INITIAL	**INITIALIZED**	**OPENING**	**OPENED**	**ERROR**
Ausgangszustand	**INITIAL**	`BUSY`	-	`PENDING`	`SUCCESS`	`FAILED`
	INITIALIZED	-	`BUSY`			
	OPENING		-	`BUSY`		

6.1.5.4.2.18 Reset

Reset

HRESULT Reset

Reset the camera controller to initial state (might require multiple calls depending on current state, until S_OK is returned)

Syntax

Definition:

```
METHOD Reset : HRESULT
```

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Methode `Reset` setzt die Kamera zurück, wenn sie im Fehlerzustand ist.

Zustandsübergänge

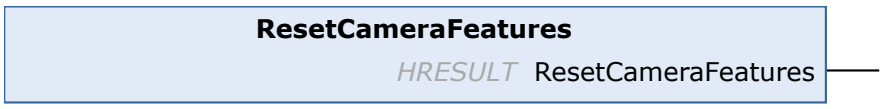
Mit `Reset` soll der Kamera-Funktionsblock vom Zustand `ERROR` in den Zustand `INITIAL` gebracht werden. Wenn das Rücksetzen erfolgreich ist, geschieht dieser Übergang sofort, ansonsten bleibt der Funktionsblock im Zustand `ERROR`.

Alle möglichen Zustandsübergänge sowie ihre Return Codes sind in folgender Übergangsmatrix dargestellt:

Return Codes (`HRESULT`)		**Ergebniszustand**	
		ERROR	**INITIAL**
Ausgangszustand	**ERROR**	`FAILED`	`SUCCESS`

Im Erfolgsfall wird S_OK oder S_FALSE (wenn kein Fehler zurückzusetzen war) als SUCCESS-Code zurückgegeben.

6.1.5.4.2.19 ResetCameraFeatures



Reset the camera features to initial state (sends the 'InitCameraCommands' defined in the GevImageAcquisition TcCOM module to the camera).

Syntax

Definition:

```
METHOD ResetCameraFeatures : HRESULT
```

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Methode `ResetCameraFeatures` setzt die Parameter der Kamera auf den Ursprungszustand zurück. Dies beinhaltet, dass die Initialization Commands [▶ 94], ohne das optionale Force-IP Kommando, neu auf die Kamera geschrieben werden.

Zustandsübergänge

Mit `ResetCameraFeatures` soll die Kamera nur ihre Parameter zurücksetzen und dabei im Zustand `OPENED` bleiben. Im Regelfall geht die Kamera jedoch kurzfristig in den Zwischenzustand `RESETTINGFEATURES`, da das Parameter-Zurücksetzen etwas Zeit braucht.

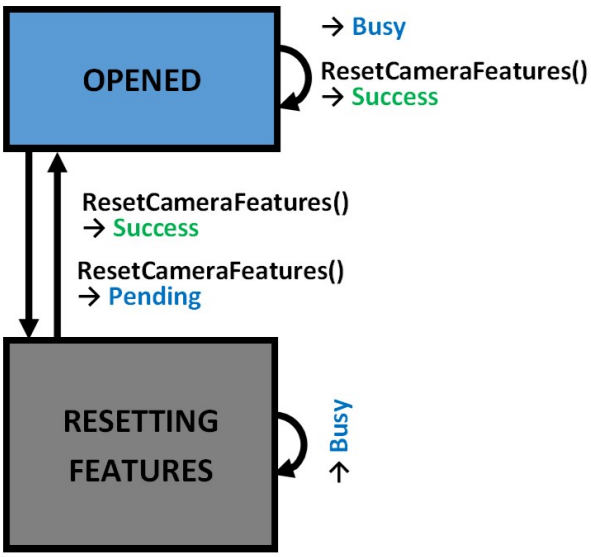
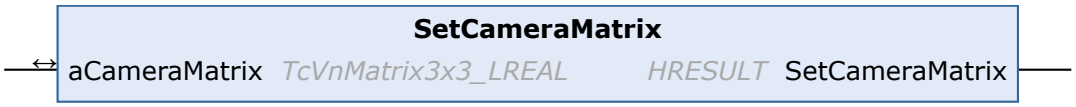


Abb. 22: Zustandsübergänge bei FB_VN_GevCameraControl.ResetCameraFeatures

Alle möglichen Zustandsübergänge sowie ihre Return Codes sind in folgender Übergangsmatrix dargestellt:

Return Codes (HRESULT)		**Ergebniszustand**		
		OPENED	**RESETTINGFEATURES**	**ERROR**
Ausgangszustand	**OPENED**	BUSY	PENDING	FAILED
	RESETTINGFEATURES	SUCCESS	BUSY	

6.1.5.4.2.20 SetCameraMatrix



Sets the camera matrix to the TcVnGevImageProvider TcCOM module

Syntax

Definition:

```
METHOD SetCameraMatrix : HRESULT
VAR_IN_OUT
    aCameraMatrix : TcVnMatrix3x3_LREAL;
END_VAR
```

In/Outputs

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 173]	The camera matrix to be copied to the TcCOM module

Return value

HRESULT [▶ 150]

6.1.5.4.2.21 SetDistortionCoefficients

Sets the distortion coefficients to the TcVnGevImageProvider TcCOM module

Syntax

Definition:

```
METHOD SetDistortionCoefficients : HRESULT
VAR_IN_OUT
    aDistortionCoefficients : TcVnArray8_LREAL;
END_VAR
```

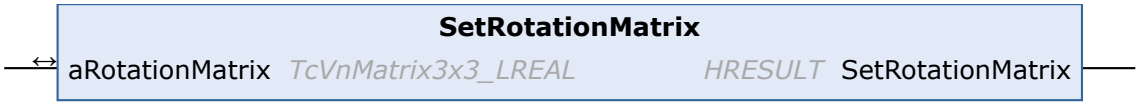
In/Outputs

Name	Type	Description
aDistortionCoefficients	TcVnArray8_LREAL [▶ 173]	The distortion coefficients to be copied to the TcCOM module

Return value

HRESULT [▶ 150]

6.1.5.4.2.22 SetRotationMatrix



Sets the rotation matrix to the TcVnGevImageProvider TcCOM module

Syntax

Definition:

```
METHOD SetRotationMatrix : HRESULT
VAR_IN_OUT
    aRotationMatrix : TcVnMatrix3x3_LREAL;
END_VAR
```

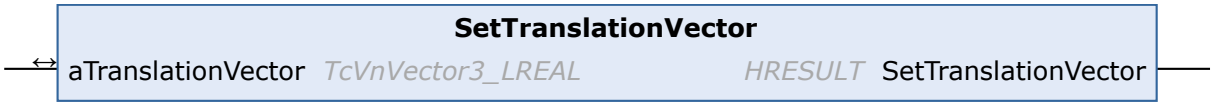
In/Outputs

Name	Type	Description
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 173]	The rotation matrix to be copied to the TcCOM module

Return value

HRESULT [▶ 150]

6.1.5.4.2.23 SetTranslationVector



Sets the translation vector to the TcVnGevImageProvider TcCOM module

Syntax

Definition:

```
METHOD SetTranslationVector : HRESULT
VAR_IN_OUT
    aTranslationVector : TcVnVector3_LREAL;
END_VAR
```

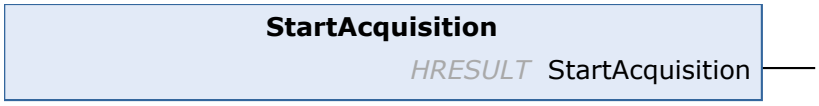
In/Outputs

Name	Type	Description
aTranslationVector	TcVnVector3_LREAL [▶ 173]	The translation vector to be copied to the TcCOM module

Return value

HRESULT [▶ 150]

6.1.5.4.2.24 StartAcquisition



Send the 'StartAcquisitionCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.

Syntax

Definition:

```
METHOD StartAcquisition : HRESULT
```

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Methode `StartAcquisition` startet die Bildaufnahme einer Kamera. Was dies im Einzelfall bedeutet, hängt von der Parametrierung der Kamera ab. Im Beispiel Bild-Akquise und Trigger [▶ 2936] sind weitere Informationen dazu zu finden. Das Starten der Bildaufnahme kann aus den Zuständen `INITIAL`, `INITIALIZED` und `OPENED` geschehen.

Wenn sich die Kamera noch in `INITIAL` oder `INITIALIZED` befindet, übernimmt die Methode `StartAcquisition` ebenfalls die Aufgaben der Methode OpenCamera [▶ 1688]. Üblicherweise wird der Zustand `OPENED` dabei nicht eingenommen.

Zustandsübergänge

Mit `StartAcquisition` soll die Kamera in den Zustand `ACQUIRING` versetzt werden. Dies kann aus den Zuständen `INITIAL`, `INITIALIZED` und `OPENED` heraus geschehen. Falls sich die Kamera in INITAL oder INITIALIZED befindet, wird zunächst der Zustandsübergang nach OPENED vollzogen. Dieser geschieht im Regelfall über den Zwischenzustand OPENING. Der Übergang von OPENED nach ACQUIRING geschieht im Regelfall über den Zwischenzustand STARTACQUISITION.

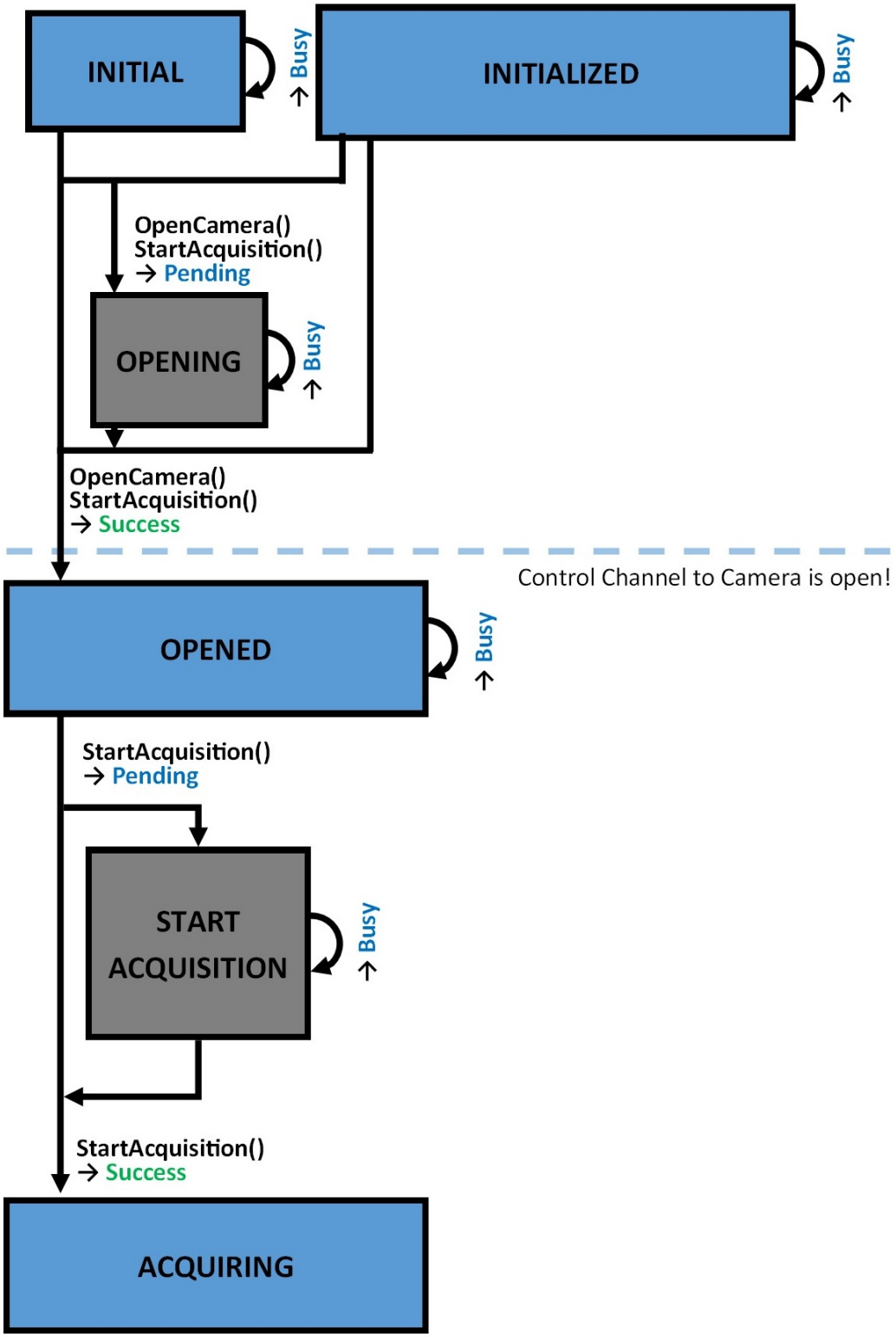


Abb. 23: Zustandsübergänge bei FB_VN_GevCameraControl.StartAcquisition

Alle möglichen Zustandsübergänge sowie ihre Return Codes sind in folgender Übergangsmatrix dargestellt:

Return Codes (HRESULT)		Ergebniszustand						
		INITIAL	INITIALIZED	OPENING	OPENED	STARTACQUISITION	ACQUIRING	ERROR
Ausgangszustand	INITIAL	BUSY	-	PENDING	SUCCESS	-	-	FAILED
	INITIALIZED	-	BUSY	PENDING				
	OPENING		-	BUSY				
	OPENED			-	BUSY	PENDING	SUCCESS	
	STARTACQUISITION				-	BUSY		

6.1.5.4.2.25 StopAcquisition

StopAcquisition

HRESULT StopAcquisition

Send the 'StopAcquisitionCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.

Syntax

Definition:

```
METHOD StopAcquisition : HRESULT
```

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Methode `StopAcquisition` stoppt die Bildaufnahme der Kamera.

Zustandsübergänge

Mit `StopAcquisition` soll die Kamera aus dem Zustand `ACQUIRING` in den Zustand `OPENED` gebracht werden. Im Regelfall geschieht dieser Übergang nicht sofort, sondern über den Zwischenzustand `STOPACQUISITION`.

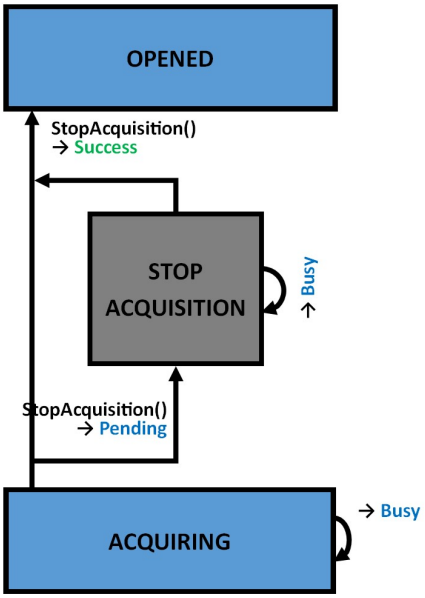
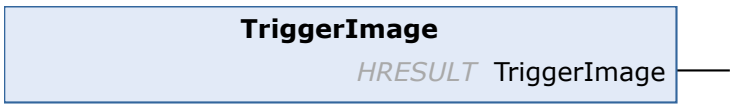


Abb. 24: Zustandsübergänge bei FB_VN_GevCameraControl.StopAcquisition

Alle möglichen Zustandsübergänge sowie ihre Return Codes sind in folgender Übergangsmatrix dargestellt:

Return Codes (HRESULT)		**Ergebniszustand**			
		ACQUIRING	**STOPACQUISITION**	**OPENED**	**ERROR**
Ausgangszustand	**ACQUIRING**	BUSY	PENDING	SUCCESS	FAILED
	STOPACQUISITION	-	BUSY		

6.1.5.4.2.26 TriggerImage



Send the 'SoftwareTriggerCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.

Syntax

Definition:

```
METHOD TriggerImage : HRESULT
```

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Zum Konfigurieren der Kamera: Bild-Akquise und Trigger [▶ 2936]

Zustandsübergänge

Mit `TriggerImage` soll die Kamera nur ein Bild aufnehmen und dabei im aktuellen Zustand `ACQUIRING` oder `OPENED` bleiben. Da die Bildaufnahme üblicherweise mehrere Zyklen dauert, geht die Kamera im Regelfall jedoch kurzzeitig in den Zwischenzustand `TRIGGERING`.

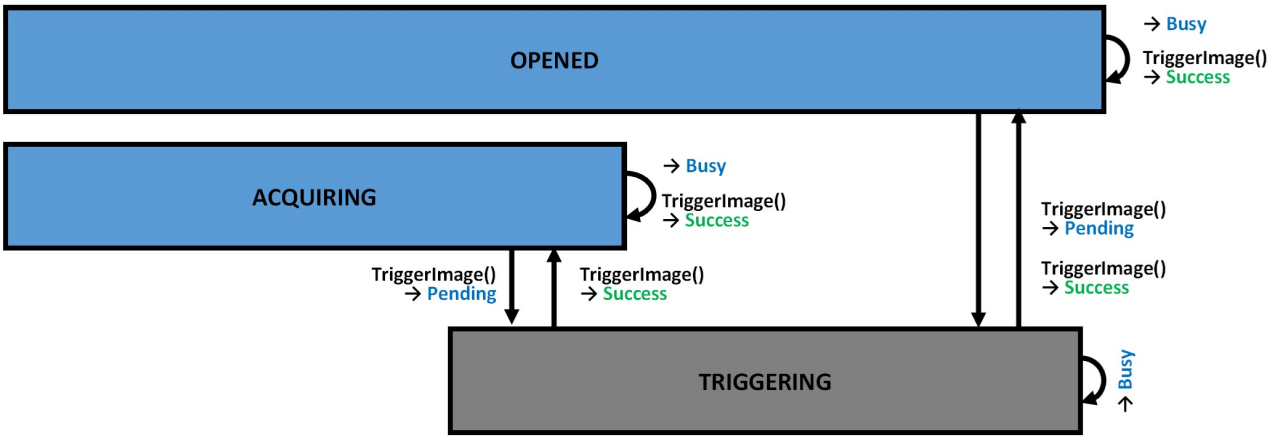


Abb. 25: Zustandübergänge bei FB_VN_GevCameraControl.TriggerImage

Alle möglichen Zustandsübergänge sowie ihre Return Codes sind in folgender Übergangsmatrix dargestellt:

Return Codes (HRESULT)		Ergebniszustand			
		OPENED	ACQUIRING	TRIGGERING	ERROR
Ausgangszustand	OPENED	BUSY	-	PENDING	FAILED
	ACQUIRING	-	BUSY		
	TRIGGERING	SUCCESS	SUCCESS	BUSY	

6.1.5.4.3 FB_VN_SimpleCameraControl

This FB provides the basic functionality to control a camera or a FileSource.

Do not call the main FB directly. Only use the available methods.

Methods

Name	Description
GetCurrentImage [▶ 1700]	Get the current available image (if any).
GetState [▶ 1701]	Get the current state of the internal camera control state machine.
Reset [▶ 1701]	Reset the camera to initial state (might require multiple calls depending on current state, until S_OK is returned)
StartAcquisition [▶ 1702]	Start the image acquisition.
StopAcquisition [▶ 1702]	Stop the image acquisition.
TriggerImage [▶ 1703]	Trigger the next image.

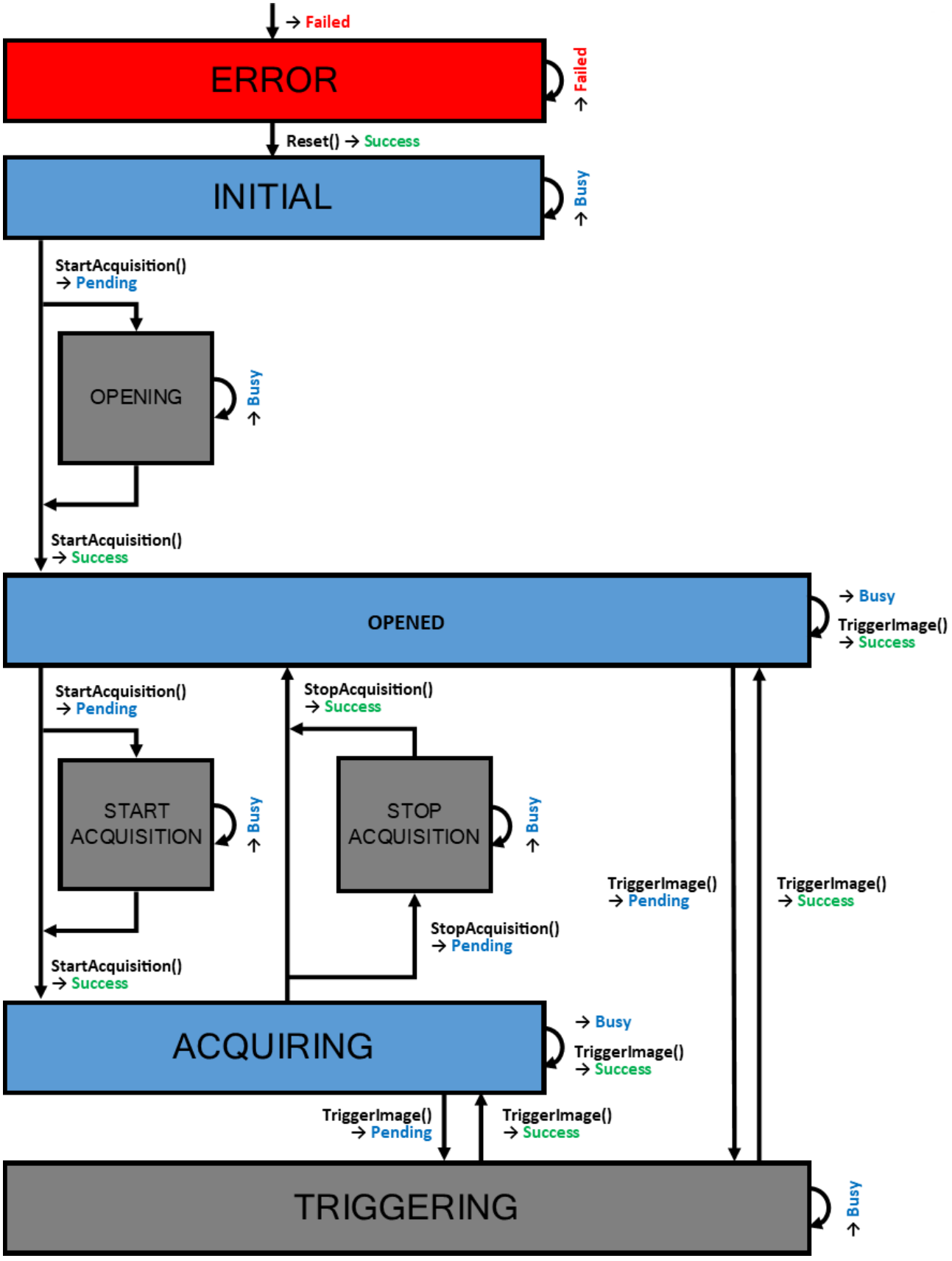
Weiterführende Informationen

Der Funktionsblock FB_VN_SimpleCameraControl vereint die Gemeinsamkeiten von FB_VN_GevCameraControl [▶ 1672] und FB_VN_FileSourceControl [▶ 1664]. Dadurch können Instanzen des Bausteins sowohl mit dem Image Provider einer Kamera als auch mit dem Image Provider eines File Source Controls verknüpft werden, wie dies auch in den First Steps [▶ 51] erfolgt.

Zustandsautomat

Die nachfolgende Grafik zeigt den Zustandsautomaten. Die Hauptzustände sind dabei blau gekennzeichnet, die Übergangszustände grau und der Error-Zustand rot. Eine Kamera nimmt, im Gegensatz zu einem File Source Control, beim Übergang von TCVN_CS_INITIAL nach TCVN_CS_ACQUIRING immer auch den Übergangszustand TCVN_CS_OPENING an und bei Bedarf auch den Hauptzustand TCVN_CS_OPENED. Wenn Init Commands vorhanden sind und der InitializationAutoMode AUTOINIT_AFTER_SO ausgewählt wurde, wird auch kurzzeitig der State INITIALIZING angenommen, in dem die automatische Initialisierung

erfolgt. Wenn ein Übergang sehr schnell erfolgt wird der entsprechende State nicht nach außen gemeldet, da innerhalb eines Zyklus schon der nachfolgende State erreicht wurde. Bei Verknüpfung mit einem File Source Control werden diese drei Zustände nicht angenommen. Die Verwendung der Trigger Image Methode aus dem `TCVN_CS_OPENED` State ist nur mit einer Kamera möglich.



Zur Steuerung des Zustandsautomaten können folgende Methoden genutzt werden:

- GetState [▶ 1701]
- StartAcquisition [▶ 1702]
- StopAcquisition [▶ 1702]
- TriggerImage [▶ 1703]
- Reset [▶ 1701]

Wenn eine Methode in einem Zustand aufgerufen wird, für den sie nicht gedacht ist, wird der Fehlercode `INVALIDSTATE` (`16#712`) zurückgegeben.

Anwendung mit kontinuierlichem Bildeinzug

Ein Zustandsautomat, um eine Kamera in den Bildaufnahme-Modus zu bringen, Fehlerzustände zu behandeln und kontinuierlich erfolgreich aufgenommene Bilder anzuzeigen, sieht z. B. so aus:

```
VAR
    hr              :   HRESULT;
    ipImageIn       :   ITcVnImage;
    ipImageInDisp   :   ITcVnDisplayableImage;
    fbCameraControl :   FB_VN_SimpleCameraControl;
    eCameraState    :   ETcVnCameraState;
    nNewImageCounter:   UINT;
END_VAR

eCameraState := fbCameraControl.GetState();

// CameraControl is in error state, so try to reset the camera connection
IF eCameraState = TCVN_CS_ERROR THEN
    hr := fbCameraControl.Reset();

// Camera not yet streaming
ELSIF eCameraState < TCVN_CS_ACQUIRING THEN
    hr := fbCameraControl.StartAcquisition();

// Camera streaming
ELSIF eCameraState = TCVN_CS_ACQUIRING THEN
    hr := fbCameraControl.GetCurrentImage(ipImageIn);

    // Check if new Image was received
    IF SUCCEEDED(hr) AND ipImageIn <> 0 THEN
        nNewImageCounter := nNewImageCounter + 1;

        // Place to call vision algorithms
        hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, hr);
    END_IF

END_IF
```

In diesem Fall kann die Kamera auf kontinuierliche Bild-Akquise [▶ 2938] oder auf einen Hardwaretrigger [▶ 2938] konfiguriert werden.

Anwendung mit Softwaretrigger

Ein Zustandsautomat, um eine Kamera in den Bildaufnahme-Modus zu bringen, Fehlerzustände zu behandeln und per Softwaretrigger eine Bildaufnahme anzutriggern, sieht z. B. so aus:

```
VAR
    hr              :   HRESULT;
    ipImageIn       :   ITcVnImage;
    ipImageInDisp   :   ITcVnDisplayableImage;
    fbCameraControl :   FB_VN_SimpleCameraControl;
    eCameraState    :   ETcVnCameraState;
    nNewImageCounter:   UINT;
    bTrigger        :   BOOL := TRUE;
END_VAR

eCameraState := fbCameraControl.GetState();

// CameraControl is in error state, so try to reset the camera connection
IF eCameraState = TCVN_CS_ERROR THEN
    hr := fbCameraControl.Reset();

// Camera trigger image
ELSIF eCameraState = TCVN_CS_TRIGGERING THEN
```

```
    hr := fbCameraControl.TriggerImage();

// Camera not yet streaming
ELSIF eCameraState < TCVN_CS_ACQUIRING THEN
    hr := fbCameraControl.StartAcquisition();

// Camera streaming
ELSIF eCameraState = TCVN_CS_ACQUIRING THEN

    IF bTrigger THEN
        hr := fbCameraControl.TriggerImage();
        IF SUCCEEDED(hr) THEN
            bTrigger := FALSE;
        END_IF
    ELSE
        hr := fbCameraControl.GetCurrentImage(ipImageIn);

        // Check if new Image was received
        IF SUCCEEDED(hr) AND ipImageIn <> 0 THEN
            nNewImageCounter := nNewImageCounter + 1;

            // Trigger next image
            bTrigger := TRUE;

            // Place to call vision algorithms
            hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, hr);
        END_IF
    END_IF

END_IF
```

In diesem Fall muss die Kamera auf einen Softwaretrigger [▶ 2938] (`TriggerSource = Software`) konfiguriert werden.

Beispiele

In vielen Funktionsbeispielen [▶ 2839] wird ein `FB_VN_SimpleCameraControl` Funktionsblock genutzt, um Beispielbilder einzuladen.

Verwandte Funktionen

- FB_VN_SimpleCameraControl [▶ 1697] für Kameras und File Sources
- FB_VN_GevCameraControl [▶ 1672] für GigE Vision Kameras
- FB_VN_FileSourceControl [▶ 1664] für File Sources

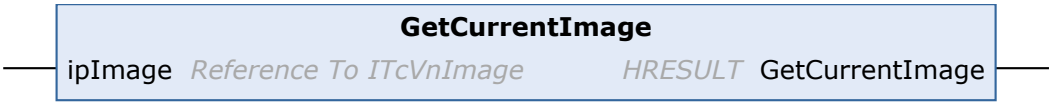
Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.4.3.1 GetCurrentImage



Get the current available image (if any).

Syntax

Definition:

```
METHOD GetCurrentImage : HRESULT
VAR_INPUT
    ipImage : Reference To ITcVnImage;
END_VAR
```

Inputs

Name	Type	Description
ipImage	Reference To ITcVnImage [▶ 426]	Returns the image

Return value

HRESULT [▶ 150]

6.1.5.4.3.2 GetState

Get the current state of the internal camera control state machine.

Syntax

Definition:

```
METHOD GetState : ETcVnCameraState
```

Return value

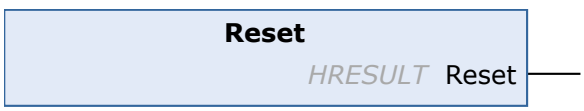
ETcVnCameraState [▶ 179]

Weiterführende Informationen

Die Methode `GetState` gibt den Zustand des Kameraobjekts bzw. File Source Objekts als Wert vom Enum ETcVnCameraState [▶ 179] zurück. Folgende Zustände sind möglich:

- ERROR
- INITIAL
- OPENING
- OPENED
- STARTACQUISITION
- STOPACQUISITION
- ACQUIRING
- TRIGGERING

6.1.5.4.3.3 Reset



Reset the camera to initial state (might require multiple calls depending on current state, until S_OK is returned)

Syntax

Definition:

```
METHOD Reset : HRESULT
```

Return value

HRESULT [▶ 150]

Weiterführende Informationen

Die Methode `Reset` setzt das Gerät zurück, wenn es im Fehlerzustand ist.

Zustandsübergänge

Mit `Reset` soll das Gerät vom Zustand `ERROR` in den Zustand `INITIAL` gebracht werden. Wenn das Rücksetzen erfolgreich ist, geschieht dieser Übergang sofort, ansonsten bleibt der Funktionsblock im Zustand `ERROR`.

Alle möglichen Zustandsübergänge sowie ihre Return Codes sind in folgender Übergangsmatrix dargestellt:

Return Codes (`HRESULT`)		**Ergebniszustand**	
		ERROR	**INITIAL**
Ausgangszustand	**ERROR**	`FAILED`	`SUCCESS`

Im Erfolgsfall wird S_OK oder S_FALSE (wenn kein Fehler zurückzusetzen war) als SUCCESS-Code zurückgegeben.

6.1.5.4.3.4 StartAcquisition

Start the image acquisition.

Syntax

Definition:

```
METHOD StartAcquisition : HRESULT
```

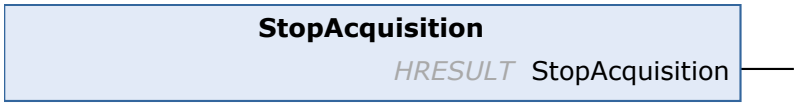
Return value

HRESULT [▶ 150]

Weiterführende Informationen

- In Verbindung mit einer Kamera entspricht diese Methode der StartAcquisition [▶ 1693]-Methode vom FB_VN_GevCameraControl [▶ 1672]
- In Verbindung mit einem File Source Control entspricht diese Methode der StartAcquisition [▶ 1668]-Methode vom FB_VN_FileSourceControl [▶ 1664]

6.1.5.4.3.5 StopAcquisition



Stop the image acquisition.

Syntax

Definition:

```
METHOD StopAcquisition : HRESULT
```

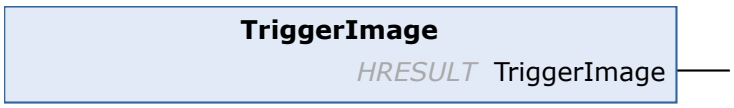
Return value

HRESULT [▶ 150]

Weiterführende Informationen

- In Verbindung mit einer Kamera entspricht diese Methode der StopAcquisition [▶ 1695]-Methode vom FB_VN_GevCameraControl [▶ 1672].
- In Verbindung mit einem File Source Control entspricht diese Methode der StopAcquisition [▶ 1669]-Methode vom FB_VN_FileSourceControl [▶ 1664].

6.1.5.4.3.6 TriggerImage



Trigger the next image.

Syntax

Definition:

```
METHOD TriggerImage : HRESULT
```

Return value

HRESULT [▶ 150]

Weiterführende Informationen

- In Verbindung mit einer Kamera entspricht diese Methode der TriggerImage [▶ 1696]-Methode vom FB_VN_GevCameraControl [▶ 1672]
- In Verbindung mit einem File Source Control entspricht diese Methode der TriggerImage [▶ 1696]-Methode vom FB_VN_FileSourceControl [▶ 1664]

6.1.5.5 Image Processing

Diese Gruppe enthält Funktionsblöcke zur Bildverarbeitung. In diesen komplexen Funktionsblöcken bleiben die einmal gesetzten Werte nach einer Ausführung erhalten. Des Weiteren können die Funktionsblöcke auch mehrere Methoden enthalten.

Die im Kapitel Functions [▶ 448] enthaltenen Funktionen haben hingegen keine internen Statusinformationen und die Variablenwerte bleiben nicht erhalten.

Funktionsblöcke

- FB_VN_GeneralizedHoughBallard [▶ 1703]
- FB_VN_SSIM [▶ 1708]

6.1.5.5.1 FB_VN_GeneralizedHoughBallard

This FB provides the Generalized Hough Ballard functionality. First, set the parameters via the corresponding methods (optional, defaults are used otherwise). Then, set the template (required). Afterwards, the Detect method can be executed.

Do not call the main FB directly. Only use the available methods.

Methods

Name	Description
Detect [▶ 1705]	Detect the template in an image (SetTemplate must have been called before).
SetCannyThreshold [▶ 1705]	Sets the canny edge detection thresholds.
SetInvAccuRatio [▶ 1706]	Sets the inverted ratio of the accumulator size in relation to the source image's size (e.g. a value of 2 means that the size is halved in both directions.)
SetLevels [▶ 1706]	Sets the number of R-table levels.
SetMinDist [▶ 1707]	Sets the minimum distance between the centers of different objects.
SetTemplate [▶ 1707]	Sets the template image to search for in the Detect method.
SetVotesThreshold [▶ 1708]	Sets the accumulator threshold, i.e. the number of votes required to detect a match (too small values lead to false detections).

Weiterführende Informationen

Der Funktionsblock `FB_VN_GeneralizedHoughBallard` findet eine beliebige Form in einem Graustufenbild mit Hilfe der verallgemeinerten Hough-Transformation. Ermittelt werden die Mittelpunktpositionen der erkannten Template-Formen, wobei unterschiedliche Skalierung und Rotation vom Template nicht berücksichtigt werden.

Parameter

Eingangsbilder

Das Template- und Eingangsbild muss ein 1-kanaliges Bild mit Element-Typ USINT (8 Bit) sein. Zuerst muss das Template-Bild mit der Methode `SetTemplate` übergeben werden. Danach kann die Ausführung mit Aufruf der `Detect` Methode erfolgen, wo dann das Eingangsbild übergeben wird.

Parameter zur Detektion

Folgende Parameter können per Methodenaufruf gesetzt werden. Ansonsten werden diese Default-Werte verwendet. Da die Parameter bereits für interne Berechnungen bei der Übergabe des Template-Bilds verwendet werden, müssen die Werte immer zuerst gesetzt werden.

CannyThresholdLow = 50

CannyThresholdHigh = 100

InvAccuRatio = 1.0

Levels = 360

MinDist = 1.0

VotesThreshold = 100

Liste der gefundenen Mittelpunkte (Rückgabewert)

Alle im Bild gefunden Mittelpunktpositionen werden nach dem Aufruf der `Detect` Methode in dem Container `ipPositions` zurückgegeben.

Anwendung

Die Ausführung besteht aus 3 Schritten. Zuerst werden die Parameter gesetzt und dann das Template-Bild übergeben. Danach reicht es aus, nur noch die `Detect` Methode mit den Bildern aufzurufen, auf denen nach dem Template gesucht werden soll. Erst wenn Parameter oder das Template geändert werden sollen, müssen die entsprechenden Schritte erneut ausgeführt werden.

```
// Set parameters first
hr := fbGenHoughBallard.SetCannyThreshold(50,150);
IF SUCCEEDED(hr) THEN hr := fbGenHoughBallard.SetInvAccuRatio(2); END_IF
IF SUCCEEDED(hr) THEN hr := fbGenHoughBallard.SetLevels(180); END_IF
```

```
IF SUCCEEDED(hr) THEN hr := fbGenHoughBallard.SetMinDist(20); END_IF
IF SUCCEEDED(hr) THEN hr := fbGenHoughBallard.SetVotesThreshold(80); END_IF

// Check if successful and set template image
IF SUCCEEDED(hr) THEN hr := fbGenHoughBallard.SetTemplate(ipTemplateImage); END_IF

// Check if successful and call detection
IF SUCCEEDED(hr) THEN hr := fbGenHoughBallard.Detect(ipSrcImage,ipPositions); END_IF
```

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.5.1.1 Detect

Detect the template in an image (SetTemplate must have been called before).

Syntax

Definition:

```
METHOD Detect : HRESULT
VAR_INPUT
    ipImage     : ITcVnImage;
    ipPositions : Reference To ITcVnContainer;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Source image (1 channel, USINT)
ipPositions	Reference To ITcVnContainer [▶ 385]	Returns the centers of the detected template positions (ContainerType_Vector_TcVnPoint2_REAL; Non-zero interface pointers are reused.)

Return value

HRESULT [▶ 150]

6.1.5.5.1.2 SetCannyThreshold

Sets the canny edge detection thresholds.

Syntax

Definition:

```
METHOD SetCannyThreshold : HRESULT
VAR_INPUT
    nLow  : DINT;
    nHigh : DINT;
END_VAR
```

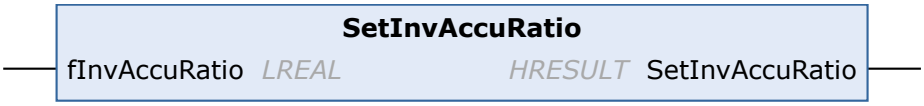
Inputs

Name	Type	Description
nLow	DINT	Low threshold (> 0)
nHigh	DINT	High threshold (> nLow, usually 2 to 3 * nLow)

Return value

HRESULT [▶ 150]

6.1.5.5.1.3 SetInvAccuRatio



Sets the inverted ratio of the accumulator size in relation to the source image's size (e.g. a value of 2 means that the size is halved in both directions.)

Syntax

Definition:

```
METHOD SetInvAccuRatio : HRESULT
VAR_INPUT
    fInvAccuRatio : LREAL;
END_VAR
```

Inputs

Name	Type	Description
fInvAccuRatio	LREAL	Inverted accumulator ratio (> 0)

Return value

HRESULT [▶ 150]

6.1.5.5.1.4 SetLevels

Sets the number of R-table levels.

Syntax

Definition:

```
METHOD SetLevels : HRESULT
VAR_INPUT
    nLevels : DINT;
END_VAR
```

Inputs

Name	Type	Description
nLevels	DINT	Number of R-table levels (> 0)

Return value

HRESULT [▶ 150]

6.1.5.5.1.5 SetMinDist

Sets the minimum distance between the centers of different objects.

Syntax

Definition:

```
METHOD SetMinDist : HRESULT
VAR_INPUT
    fMinDist : LREAL;
END_VAR
```

Inputs

Name	Type	Description
fMinDist	LREAL	Minimum distance (> 0)

Return value

HRESULT [▶ 150]

6.1.5.5.1.6 SetTemplate

Sets the template image to search for in the Detect method.

Syntax

Definition:

```
METHOD SetTemplate : HRESULT
VAR_INPUT
    ipImage : ITcVnImage;
END_VAR
```

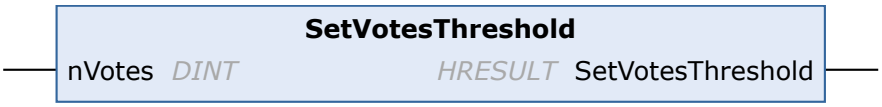
Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Template image (1 channel, USINT)

Return value

HRESULT [▶ 150]

6.1.5.5.1.7 SetVotesThreshold



Sets the accumulator threshold, i.e. the number of votes required to detect a match (too small values lead to false detections).

Syntax

Definition:

```
METHOD SetVotesThreshold : HRESULT
VAR_INPUT
    nVotes : DINT;
END_VAR
```

Inputs

Name	Type	Description
nVotes	DINT	Number of votes required to detect a match (> 0)

Return value

HRESULT [▶ 150]

6.1.5.5.2 FB_VN_SSIM

This FB provides the structural similarity (SSIM) functionality. First, set the reference image. Afterwards, the Compute method can be executed.

Do not call the main FB directly. Only use the available methods.

Methods

Name	Description
Compute [▶ 1709]	Compute the structural similarity (SetReference must have been called before).
SetReference [▶ 1710]	Sets the reference image, used by the SSIM method.

Weiterführende Informationen

Der Funktionsblock `FB_VN_SSIM` berechnet die strukturelle Ähnlichkeit von zwei Graustufenbildern. Als Referenz wird von einem störungsfreien Bild ausgegangen.

Parameter

Eingangsbilder

Das Referenz- und Eingangsbild muss ein 1-kanaliges Bild mit Element-Typ `USINT` (8 Bit) sein. Zuerst muss das Referenz-Bild mit der Methode `SetReference` übergeben werden. Danach kann die Ausführung mit Aufruf der `Compute` Methode erfolgen, der das Eingangsbild übergeben wird.

Ausgangsbild (Rückgabewert)

Das Ausgangsbild ist vom Typ `Real` und auf den Wertebereich von 0-1 skaliert. Eine 1 bedeutet Übereinstimmung und 0 eine maximale Abweichung.

Anwendung

Die Ausführung besteht aus 2 Schritten. Zuerst muss das Referenz-Bild gesetzt werden. Danach kann die `Compute` Methode aufgerufen werden, der das aktuell zu vergleichende Eingangsbild übergeben wird.

Zur weiteren Analyse des Ergebnisbilds z. B. mit der F_VN_DetectBlobs [▶ 1207] Funktion ist es notwendig, das Bild in ein `USINT` zu konvertieren. Dazu kann die Funktion F_VN_ConvertElementTypeExp [▶ 799] verwendet werden. Über die Funktionen F_VN_ImageAverage [▶ 1247] oder F_VN_MinPixelValue [▶ 1258] kann direkt auf Basis des Ergebnisbilds ein Übereinstimmungswert ermittelt werden.

```
// Set reference image first
hr := fbSSIM.SetReference(ipImage := ipRefImage);

// Check if successful, set imput image and compute the result
IF SUCCEEDED(hr) and ipRefImage <> 0 THEN
    hr := fbSSIM.Compute(ipSrcImage:= ipImageIn, ipSSIM:= ipResultImage);
    // Convert and scale the result image to USINT
    hr := F_VN_ConvertElementTypeExp(ipResultImage, ipResult2Image, ETcVnElementType.TCVN_ET_USINT,
(255.0/1.0), 0, hr);
END_IF
```

Hohe Speicherauslastung bei großen Bildern

Da die Funktion im Hintergrund viele Berechnungen mit Real-Werten durchführt, wird in Abhängigkeit zur Bildgröße entsprechend mehr Router Speicher und Rechenleistung benötigt.

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.1.5.5.2.1 Compute

Compute the structural similarity (SetReference must have been called before).

Syntax

Definition:

```
METHOD Compute : HRESULT
VAR_INPUT
    ipSrcImage : ITcVnImage;
    ipSSIM     : Reference To ITcVnImage;
END_VAR
```

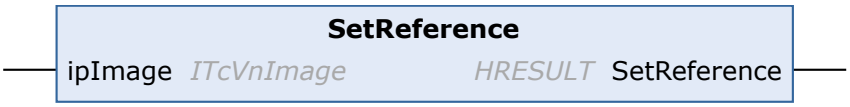
Inputs

Name	Type	Description
ipSrcImage	ITcVnImage [▶ 426]	Source image (USINT)
ipSSIM	Reference To ITcVnImage [▶ 426]	Returns the SSIM (REAL, values from 0 to 1)

Return value

HRESULT [▶ 150]

6.1.5.5.2.2 SetReference



Sets the reference image, used by the SSIM method.

Syntax

Definition:

```
METHOD SetReference : HRESULT
VAR_INPUT
    ipImage : ITcVnImage;
END_VAR
```

Inputs

Name	Type	Description
ipImage	ITcVnImage [▶ 426]	Reference image (USINT)

Return value

HRESULT [▶ 150]

6.2 C++

Diese Kapitel beinhaltet die TwinCAT Vision API-Beschreibung zur programmatischen Verwendung der Bibliothek in C++.

- Datentypen [▶ 1710]
  (Aliase, Konstanten, Arrays, Enums, Strukturen)
- Interfaces [▶ 1783]
  (Bilder, Container, etc.)
- Funktionen [▶ 1944]
  (Bildverarbeitung)
- Funktionsblöcke [▶ 2732]
  (Kommunikation mit Kameras und dem Dateisystem, Bildverarbeitung)

6.2.1 Data Types

Zu den Datentypen gehören vier Typen, die jeweils das Präfix „TcVn“ enthalten:

- Alias-Namen [▶ 1711] für bestimmte Array-Typen geben Bezug zur Bildverarbeitung
- Arrays [▶ 1713] mit einer bestimmten Kombination von Datentypen und definierter Anzahl, die häufig von den Bildverarbeitungsfunktionen verwendet werden.
- Enums [▶ 1713] helfen vor allem bei der Parametrierung von Funktionen und enthalten das zusätzliche Präfix „E“
- Strukturen [▶ 1763] kapseln mehrere zusammenhängende Informationen (bspw. für geometrische Beschreibungen)

Außerdem sind Konstanten [▶ 1711] zur Beschreibung von Container-Typen vorhanden.

6.2.1.1 Aliases

Name	Type
GVCP_REGISTER_ADDRESS	ULONG
TcVnPoint	TcVnPoint2_DINT [▶ 1711]
TcVnPoint2_DINT	TcVnVector2_DINT [▶ 1713]
TcVnPoint2_LREAL	TcVnVector2_LREAL [▶ 1713]
TcVnPoint2_REAL	TcVnVector2_REAL [▶ 1713]
TcVnPoint3_LREAL	TcVnVector3_LREAL [▶ 1713]
TcVnPoint3_REAL	TcVnVector3_REAL [▶ 1713]
TcVnRectangle	TcVnRectangle_DINT [▶ 1781]

6.2.1.2 Constants

Die folgenden Konstanten stellen Typen von Containern [▶ 163] als `GUID` dar. Diese werden z. B. bei Funktionen wie CreateContainer [▶ 2209] benötigt.

Name	Type
ContainerType_Vector_SINT	GUID
ContainerType_Vector_USINT	GUID
ContainerType_Vector_INT	GUID
ContainerType_Vector_UINT	GUID
ContainerType_Vector_DINT	GUID
ContainerType_Vector_UDINT	GUID
ContainerType_Vector_ULINT	GUID
ContainerType_Vector_REAL	GUID
ContainerType_Vector_LREAL	GUID
ContainerType_Vector_ITcVnImage	GUID
ContainerType_Vector_TcVnPoint2_DINT	GUID
ContainerType_Vector_TcVnPoint2_REAL	GUID
ContainerType_Vector_TcVnPoint2_LREAL	GUID
ContainerType_Vector_TcVnPoint3_REAL	GUID
ContainerType_Vector_TcVnPoint3_LREAL	GUID
ContainerType_Vector_Vector_TcVnPoint2_DINT	GUID
ContainerType_Vector_Vector_TcVnPoint2_REAL	GUID
ContainerType_Vector_Vector_TcVnPoint3_REAL	GUID
ContainerType_Vector_Vector_UDINT	GUID
ContainerType_Vector_Vector_ULINT	GUID
ContainerType_Vector_Vector_REAL	GUID
ContainerType_Vector_Vector_LREAL	GUID
ContainerType_Vector_TcVnVector2_SINT	GUID
ContainerType_Vector_TcVnVector2_USINT	GUID
ContainerType_Vector_TcVnVector2_INT	GUID
ContainerType_Vector_TcVnVector2_UINT	GUID
ContainerType_Vector_TcVnVector2_DINT	GUID
ContainerType_Vector_TcVnVector2_REAL	GUID
ContainerType_Vector_TcVnVector3_SINT	GUID
ContainerType_Vector_TcVnVector3_USINT	GUID
ContainerType_Vector_TcVnVector3_INT	GUID
ContainerType_Vector_TcVnVector3_UINT	GUID
ContainerType_Vector_TcVnVector3_REAL	GUID
ContainerType_Vector_TcVnVector4_SINT	GUID
ContainerType_Vector_TcVnVector4_USINT	GUID
ContainerType_Vector_TcVnVector4_INT	GUID
ContainerType_Vector_TcVnVector4_UINT	GUID
ContainerType_Vector_TcVnVector4_DINT	GUID
ContainerType_Vector_TcVnVector4_REAL	GUID
ContainerType_Vector_TcVnVector4_LREAL	GUID
ContainerType_Vector_TcVnRectangle_DINT	GUID
ContainerType_Vector_TcVnRotatedRectangle	GUID
ContainerType_Vector_TcVnKeyPoint	GUID
ContainerType_Vector_TcVnDMatch	GUID
ContainerType_Vector_Vector_TcVnDMatch	GUID
ContainerType_String_SINT	GUID
ContainerType_Vector_String_SINT	GUID

6.2.1.3 Arrays

Name	Type
TcVnArray10_ITcVnImage	ITcVnImage* [▶ 1925][10]
TcVnArray3_Point2_REAL	TcVnPoint2_REAL [▶ 1711][3]
TcVnArray33_UDINT	ULONG[33]
TcVnArray4_LREAL	double[4]
TcVnArray4_Point2_REAL	TcVnPoint2_REAL [▶ 1711][4]
TcVnArray7_LREAL	double[7]
TcVnArray8_LREAL	double[8]
TcVnMatrix2x3_LREAL	double[2][3]
TcVnMatrix3x3_LREAL	double[3][3]
TcVnVector2_DINT	LONG[2]
TcVnVector2_INT	SHORT[2]
TcVnVector2_LREAL	double[2]
TcVnVector2_REAL	float[2]
TcVnVector2_SINT	char[2]
TcVnVector2_UINT	USHORT[2]
TcVnVector2_USINT	unsigned char[2]
TcVnVector3_INT	SHORT[3]
TcVnVector3_LREAL	double[3]
TcVnVector3_REAL	float[3]
TcVnVector3_SINT	char[3]
TcVnVector3_UINT	USHORT[3]
TcVnVector3_USINT	unsigned char[3]
TcVnVector4_DINT	LONG[4]
TcVnVector4_INT	SHORT[4]
TcVnVector4_LREAL	double[4]
TcVnVector4_REAL	float[4]
TcVnVector4_SINT	char[4]
TcVnVector4_UINT	USHORT[4]
TcVnVector4_USINT	unsigned char[4]

6.2.1.4 Enums

6.2.1.4.1 ETcVn2dCodeSearchStrategy

Offers search strategies for 2d code reading (multiple TCVN_CSS_XXX_INVERTED or multiple TCVN_CSS_XXX_FLIPPED cannot be combined).

Syntax

Definition:

```
enum ETcVn2dCodeSearchStrategy : ULONG
{
    CSS_DEFAULT            = 1,
    CSS_ONLY_NOT_INVERTED  = 8,
    CSS_FIRST_NOT_INVERTED = 10,
    CSS_ONLY_INVERTED      = 12,
    CSS_FIRST_INVERTED     = 14,
    CSS_ONLY_NOT_FLIPPED   = 64,
    CSS_FIRST_NOT_FLIPPED  = 80,
    CSS_ONLY_FLIPPED       = 96,
    CSS_FIRST_FLIPPED      = 112
};
```

Values

Name	Description
CSS_DEFAULT	The algorithm searches for codes using the default searching strategy, all other flags are ignored (dependant on code type).
CSS_ONLY_NOT_INVERTED	The algorithm searches for codes only in the non-inverted image.
CSS_FIRST_NOT_INVERTED	The algorithm first searches for codes in the non-inverted and then in the inverted image.
CSS_ONLY_INVERTED	The algorithm searches for codes only in the inverted image.
CSS_FIRST_INVERTED	The algorithm first searches for codes in the inverted and then in the non-inverted image.
CSS_ONLY_NOT_FLIPPED	The algorithm searches for codes only in the non-mirrored image.
CSS_FIRST_NOT_FLIPPED	The algorithm first searches for codes in the non-mirrored and then in the mirrored image.
CSS_ONLY_FLIPPED	The algorithm searches for codes only in the mirrored image.
CSS_FIRST_FLIPPED	The algorithm first searches for codes in the mirrored and then in the non-mirrored image.

6.2.1.4.2 ETcVnAdaptiveThresholdMethod

Offers methods for adaptive threshold.

Syntax

Definition:

```
enum ETcVnAdaptiveThresholdMethod : LONG
{
    ATM_MEAN     = 0,
    ATM_GAUSSIAN = 1
};
```

Values

Name	Description
ATM_MEAN	The threshold value is calculated as the mean of the nBlockSize x nBlockSize neighborhood of (x,y) minus fConstant.
ATM_GAUSSIAN	The threshold value is the weighted sum (cross-correlation with a Gaussian window) of the nBlockSize x nBlockSize neighborhood of (x,y) minus fConstant.

6.2.1.4.3 ETcVnBarcodeSearchDirection

Offers search directions for linear barcodes.

Syntax

Definition:

```
enum ETcVnBarcodeSearchDirection : ULONG
{
    BSD_ANY        = 0,
    BSD_HORIZONTAL = 1,
    BSD_VERTICAL   = 2
};
```

Values

Name	Description
BSD_ANY	The algorithm searches for codes in any supported directions.
BSD_HORIZONTAL	The algorithm searches for codes in horizontal direction.
BSD_VERTICAL	The algorithm searches for codes in vertical direction.

6.2.1.4.4 ETcVnBarcodeType

Offers linear barcode types to search for.

Syntax

Definition:

```
enum ETcVnBarcodeType : ULONG
{
    BT_CODABAR        = 4,
    BT_CODE39         = 8,
    BT_CODE93         = 16,
    BT_CODE128        = 32,
    BT_EAN8           = 128,
    BT_EAN13          = 256,
    BT_ITF            = 512,
    BT_UPCA           = 32768,
    BT_UPCE           = 65536,
    BT_ANY            = 99260,
    BT_CODE39EXTENDED = 262144
};
```

Values

Name	Description
BT_CODABAR	The algorithm searches for Codabar codes.
BT_CODE39	The algorithm searches for Code-39 codes.
BT_CODE93	The algorithm searches for Code-93 codes.
BT_CODE128	The algorithm searches for Code-128 codes.
BT_EAN8	The algorithm searches for EAN-8 codes.
BT_EAN13	The algorithm searches for EAN-13 codes.
BT_ITF	The algorithm searches for ITF codes.
BT_UPCA	The algorithm searches for UPC-A codes.
BT_UPCE	The algorithm searches for UPC-E codes.
BT_ANY	The algorithm searches for any supported linear barcode. If the type of the code is known, it is recommended to select the specific type directly.
BT_CODE39EXTENDED	The algorithm searches for Code-93-Extended codes.

6.2.1.4.5 ETcVnBlobCombination

Offers multiple blob contours to choose from, which should be returned in a multi-threshold scenario (used in TcVnParamsBlobDetection, which in return is used in F_VN_DetectBlobs).

Syntax

Definition:

```
enum ETcVnBlobCombination : LONG
{
    BC_SMALLEST         = 0,
    BC_LARGEST          = 1,
```

```
    BC_MIN_THRESHOLD    = 2,
    BC_MAX_THRESHOLD    = 3,
    BC_MEDIAN_THRESHOLD = 4
};
```

Values

Name	Description
BC_SMALLEST	Returns the smallest blob contour.
BC_LARGEST	Returns the largest blob contour.
BC_MIN_THRESHOLD	Returns the blob contour for the minimum threshold.
BC_MAX_THRESHOLD	Returns the blob contour for the maximum threshold.
BC_MEDIAN_THRESHOLD	Returns the blob contour for the median threshold.

6.2.1.4.6 ETcVnBoostClassifierType

Offers different types of Boost classifiers.

Syntax

Definition:

```
enum ETcVnBoostClassifierType : LONG
{
    BCT_DISCRETE = 0,
    BCT_REAL     = 1,
    BCT_LOGIT    = 2,
    BCT_GENTLE   = 3
};
```

Values

Name	Description
BCT_DISCRETE	Discrete AdaBoost
BCT_REAL	Real AdaBoost
BCT_LOGIT	LogitBoost
BCT_GENTLE	Gentle AdaBoost

6.2.1.4.7 ETcVnBorderInterpolationMethod

Offers methods to extrapolate values of non-existing pixels. On the one hand, this is used for filtering functions to enable filtering at the image borders (where the filter mask reaches over the border). On the other hand, this is used to extrapolate the undefined pixels after a geometric image transformation.

Syntax

Definition:

```
enum ETcVnBorderInterpolationMethod : LONG
{
    BIM_CONSTANT             = 0,
    BIM_REPLICATE            = 1,
    BIM_REFLECT              = 2,
    BIM_WRAP                 = 3,
    BIM_REFLECT_101          = 4,
    BIM_DEFAULT              = 4,
    BIM_TRANSPARENT          = 5,
    BIM_ISOLATED_CONSTANT    = 16,
    BIM_ISOLATED_REPLICATE   = 17,
    BIM_ISOLATED_REFLECT     = 18,
    BIM_ISOLATED_WRAP        = 19,
    BIM_ISOLATED_REFLECT_101 = 20
};
```

Values

Name	Description
BIM_CONSTANT	iiiiiii | abcdefgh | iiiiiii (with some specified i)
BIM_REPLICATE	aaaaaaa | abcdefgh | hhhhhhh
BIM_REFLECT	gfedcba | abcdefgh | hgfedcb
BIM_WRAP	bcdefgh | abcdefgh | abcdefg
BIM_REFLECT_101	hgfedcb | abcdefgh | gfedcba
BIM_DEFAULT	Choose this if you don't know which method to select (same as REFLECT_101).
BIM_TRANSPARENT	Corresponding pixels in the destination image will not be modified. Only available for geometric image transformations.
BIM_ISOLATED_CONSTANT	Similar to CONSTANT, but ROIs are handled as isolated, which means surrounding image pixels outside the ROI are ignored.
BIM_ISOLATED_REPLICATE	Similar to REPLICATE, but ROIs are handled as isolated, which means surrounding image pixels outside the ROI are ignored.
BIM_ISOLATED_REFLECT	Similar to REFLECT, but ROIs are handled as isolated, which means surrounding image pixels outside the ROI are ignored.
BIM_ISOLATED_WRAP	Similar to WRAP, but ROIs are handled as isolated, which means surrounding image pixels outside the ROI are ignored.
BIM_ISOLATED_REFLECT_101	Similar to REFLECT_101, but ROIs are handled as isolated, which means surrounding image pixels outside the ROI are ignored.

6.2.1.4.8 ETcVnCalibrationPattern

Offers common calibration pattern types.

Syntax

Definition:

```
enum ETcVnCalibrationPattern : LONG
{
    CP_CIRCLES_SYM         = 0,
    CP_CIRCLES_ASYM        = 1,
    CP_CIRCLES_ASYM_INDENT = 2
};
```

Values

Name	Description
CP_CIRCLES_SYM	Symmetric circle pattern
CP_CIRCLES_ASYM	Asymmetric circle pattern, i.e. every 2nd row is indented, starting unindented
CP_CIRCLES_ASYM_INDENT	Asymmetric circle pattern, i.e. every 2nd row is indented, starting indented

6.2.1.4.9 ETcVnCalibrationPatternOrigin

Offers calibration pattern origins.

Syntax

Definition:

```
enum ETcVnCalibrationPatternOrigin : LONG
{
    CPO_TOPLEFT      = 0,
    CPO_TOPCENTER    = 1,
    CPO_TOPRIGHT     = 2,
    CPO_CENTERLEFT   = 3,
    CPO_CENTER       = 4,
    CPO_CENTERRIGHT  = 5,
    CPO_BOTTOMLEFT   = 6,
    CPO_BOTTOMCENTER = 7,
    CPO_BOTTOMRIGHT  = 8
};
```

Values

Name	Description
CPO_TOPLEFT	Origin is at top left
CPO_TOPCENTER	Origin is at top center
CPO_TOPRIGHT	Origin is at top right
CPO_CENTERLEFT	Origin is at center left
CPO_CENTER	Origin is at center
CPO_CENTERRIGHT	Origin is at center right
CPO_BOTTOMLEFT	Origin is at bottom left
CPO_BOTTOMCENTER	Origin is at bottom center
CPO_BOTTOMRIGHT	Origin is at bottom right

6.2.1.4.10 ETcVnCameraState

Specifies the state of a camera controller, which controls an attached camera.

Syntax

Definition:

```
enum ETcVnCameraState : LONG
{
    CS_ERROR             = -1,
    CS_INITIAL           = 0,
    CS_INITIALIZING      = 1,
    CS_INITIALIZED       = 2,
    CS_OPENING           = 3,
    CS_OPENED            = 4,
    CS_STARTACQUISITION  = 5,
    CS_ACQUIRING         = 6,
    CS_STOPACQUISITION   = 7,
    CS_RESETTINGFEATURES = 8,
    CS_TRIGGERING        = 9,
    CS_CLOSING           = 10
};
```

Values

Name	Description
CS_ERROR	The camera controller is in an error state (e.g. the connection to the camera was lost).
CS_INITIAL	The camera controller is in the initial state and ready to establish the connection to the camera.
CS_INITIALIZING	The camera controller is initializing the camera.
CS_INITIALIZED	The camera controller has initialized the camera.
CS_OPENING	The camera controller is establishing the connection to the camera.
CS_OPENED	The connection to the camera has been established and the camera controller is ready to start the image acquisition.
CS_STARTACQUISITION	The camera controller is starting the image acquisition.
CS_ACQUIRING	The camera is sending images (either streaming or manually triggered, depending on the configuration).
CS_STOPACQUISITION	The camera controller is stopping the image acquisition.
CS_RESETTINGFEATURES	The camera controller is resetting the features on the camera.
CS_TRIGGERING	The camera controller is processing a softwaretrigger.
CS_CLOSING	The camera controller is closing the connection to the camera.

6.2.1.4.11 ETcVnClusteringAlgorithm

Offers clustering algorithms

Syntax

Definition:

```
enum ETcVnClusteringAlgorithm : LONG
{
    CA_KMEANSPP = 0,
    CA_LBG      = 1
};
```

Values

Name	Description
CA_KMEANSPP	KMeans++ (fixed number of clusters)
CA_LBG	LBG variant (dynamic number of clusters)

6.2.1.4.12 ETcVnColorMap

Offers color maps (similar to GNU Octave/MATLAB types).

Syntax

Definition:

```
enum ETcVnColorMap : LONG
{
    CM_AUTUMN           = 0,
    CM_BONE             = 1,
    CM_JET              = 2,
    CM_WINTER           = 3,
```

```
    CM_RAINBOW          = 4,
    CM_OCEAN            = 5,
    CM_SUMMER           = 6,
    CM_SPRING           = 7,
    CM_COOL             = 8,
    CM_HSV              = 9,
    CM_PINK             = 10,
    CM_HOT              = 11,
    CM_PARULA           = 12,
    CM_MAGMA            = 13,
    CM_INFERNO          = 14,
    CM_PLASMA           = 15,
    CM_VIRIDIS          = 16,
    CM_CIVIDIS          = 17,
    CM_TWILIGHT         = 18,
    CM_TWILIGHT_SHIFTED = 19,
    CM_TURBO            = 20
};
```

Values

Name	Description
CM_AUTUMN	Red - orange - yellow
CM_BONE	Black - blueish gray - white
CM_JET	Dark blue - green - dark red
CM_WINTER	Blue - green
CM_RAINBOW	Red - green - purple
CM_OCEAN	Black - blue - white
CM_SUMMER	Green - yellow
CM_SPRING	Pink - yellow
CM_COOL	Cyan - magenta
CM_HSV	Red - green - blue - red
CM_PINK	Black - pink - light yellow - white
CM_HOT	Black - red - yellow - white
CM_PARULA	Blue - yellow
CM_MAGMA	Black - magenta - light yellow
CM_INFERNO	Black - magenta - yellow
CM_PLASMA	Blue - magenta - yellow
CM_VIRIDIS	Purple - yellow
CM_CIVIDIS	Blue - gray - yellow
CM_TWILIGHT	White - light blue - dark purple - light red - white
CM_TWILIGHT_SHIFTED	Dark purple - light blue - white - light red - dark purple
CM_TURBO	Dark blue - light green - dark red

6.2.1.4.13 ETcVnColorMapSize

Offers color map sizes.

Syntax

Definition:

```
enum ETcVnColorMapSize : ULONG
{
    CMS_256   = 256,
    CMS_65536 = 65536
};
```

Values

Name	Description
CMS_256	256 colors, used for 8 bit images (TCVN_ET_USINT)
CMS_65536	65536 colors, used for 16 bit images (TCVN_ET_UINT)

6.2.1.4.14 ETcVnColorSpaceTransform

Offers color space transformations.

Syntax

Definition:

```
enum ETcVnColorSpaceTransform : LONG
{
    CST_BGR_TO_BGRA                 = 0,
    CST_RGB_TO_RGBA                 = 0,
    CST_BGRA_TO_BGR                 = 1,
    CST_RGBA_TO_RGB                 = 1,
    CST_BGR_TO_RGBA                 = 2,
    CST_RGB_TO_BGRA                 = 2,
    CST_BGRA_TO_RGB                 = 3,
    CST_RGBA_TO_BGR                 = 3,
    CST_BGR_TO_RGB                  = 4,
    CST_RGB_TO_BGR                  = 4,
    CST_BGRA_TO_RGBA                = 5,
    CST_RGBA_TO_BGRA                = 5,
    CST_BGR_TO_GRAY                 = 6,
    CST_RGB_TO_GRAY                 = 7,
    CST_GRAY_TO_BGR                 = 8,
    CST_GRAY_TO_RGB                 = 8,
    CST_GRAY_TO_BGRA                = 9,
    CST_GRAY_TO_RGBA                = 9,
    CST_BGRA_TO_GRAY                = 10,
    CST_RGBA_TO_GRAY                = 11,
    CST_RGB_TO_BGR_565              = 12,
    CST_BGR_TO_BGR_565              = 13,
    CST_BGR_565_TO_RGB              = 14,
    CST_BGR_565_TO_BGR              = 15,
    CST_RGBA_TO_BGR_565             = 16,
    CST_BGRA_TO_BGR_565             = 17,
    CST_BGR_565_TO_RGBA             = 18,
    CST_BGR_565_TO_BGRA             = 19,
    CST_GRAY_TO_BGR_565             = 20,
    CST_BGR_565_TO_GRAY             = 21,
    CST_RGB_TO_BGR_555              = 22,
    CST_BGR_TO_BGR_555              = 23,
    CST_BGR_555_TO_RGB              = 24,
    CST_BGR_555_TO_BGR              = 25,
    CST_RGBA_TO_BGR_555             = 26,
    CST_BGRA_TO_BGR_555             = 27,
    CST_BGR_555_TO_RGBA             = 28,
    CST_BGR_555_TO_BGRA             = 29,
    CST_GRAY_TO_BGR_555             = 30,
    CST_BGR_555_TO_GRAY             = 31,
    CST_BGR_TO_XYZ                  = 32,
    CST_RGB_TO_XYZ                  = 33,
    CST_XYZ_TO_BGR                  = 34,
    CST_XYZ_TO_RGB                  = 35,
    CST_BGR_TO_YCRCB                = 36,
    CST_RGB_TO_YCRCB                = 37,
    CST_YCRCB_TO_BGR                = 38,
    CST_YCRCB_TO_RGB                = 39,
    CST_BGR_TO_HSV                  = 40,
    CST_RGB_TO_HSV                  = 41,
    CST_BGR_TO_LAB                  = 44,
    CST_RGB_TO_LAB                  = 45,
    CST_BAYER_RG_TO_BGR             = 46,
    CST_BAYER_BG_TO_RGB             = 46,
    CST_BAYER_GR_TO_BGR             = 47,
    CST_BAYER_GB_TO_RGB             = 47,
    CST_BAYER_BG_TO_BGR             = 48,
    CST_BAYER_RG_TO_RGB             = 48,
```

```
    CST_BAYER_GB_TO_BGR             = 49,
    CST_BAYER_GR_TO_RGB             = 49,
    CST_BGR_TO_LUV                  = 50,
    CST_RGB_TO_LUV                  = 51,
    CST_BGR_TO_HLS                  = 52,
    CST_RGB_TO_HLS                  = 53,
    CST_HSV_TO_BGR                  = 54,
    CST_HSV_TO_RGB                  = 55,
    CST_LAB_TO_BGR                  = 56,
    CST_LAB_TO_RGB                  = 57,
    CST_LUV_TO_BGR                  = 58,
    CST_LUV_TO_RGB                  = 59,
    CST_HLS_TO_BGR                  = 60,
    CST_HLS_TO_RGB                  = 61,
    CST_BAYER_RG_TO_BGR_VNG         = 62,
    CST_BAYER_GR_TO_BGR_VNG         = 63,
    CST_BAYER_BG_TO_BGR_VNG         = 64,
    CST_BAYER_GB_TO_BGR_VNG         = 65,
    CST_BAYER_RG_TO_RGB_VNG         = 64,
    CST_BAYER_GR_TO_RGB_VNG         = 65,
    CST_BAYER_BG_TO_RGB_VNG         = 62,
    CST_BAYER_GB_TO_RGB_VNG         = 63,
    CST_BGR_TO_HSV_FULL             = 66,
    CST_RGB_TO_HSV_FULL             = 67,
    CST_BGR_TO_HLS_FULL             = 68,
    CST_RGB_TO_HLS_FULL             = 69,
    CST_HSV_TO_BGR_FULL             = 70,
    CST_HSV_TO_RGB_FULL             = 71,
    CST_HLS_TO_BGR_FULL             = 72,
    CST_HLS_TO_RGB_FULL             = 73,
    CST_LBGR_TO_LAB                 = 74,
    CST_LRGB_TO_LAB                 = 75,
    CST_LBGR_TO_LUV                 = 76,
    CST_LRGB_TO_LUV                 = 77,
    CST_LAB_TO_LBGR                 = 78,
    CST_LAB_TO_LRGB                 = 79,
    CST_LUV_TO_LBGR                 = 80,
    CST_LUV_TO_LRGB                 = 81,
    CST_BGR_TO_YUV                  = 82,
    CST_RGB_TO_YUV                  = 83,
    CST_YUV_TO_BGR                  = 84,
    CST_YUV_TO_RGB                  = 85,
    CST_BAYER_RG_TO_GRAY            = 86,
    CST_BAYER_GR_TO_GRAY            = 87,
    CST_BAYER_BG_TO_GRAY            = 88,
    CST_BAYER_GB_TO_GRAY            = 89,
    CST_YUV_420_NV12_TO_RGB         = 90,
    CST_YUV_420_NV12_TO_BGR         = 91,
    CST_YUV_420_NV21_TO_RGB         = 92,
    CST_YUV_420_NV21_TO_BGR         = 93,
    CST_YUV_420_SP_TO_RGB           = 92,
    CST_YUV_420_SP_TO_BGR           = 93,
    CST_YUV_420_NV12_TO_RGBA        = 94,
    CST_YUV_420_NV12_TO_BGRA        = 95,
    CST_YUV_420_NV21_TO_RGBA        = 96,
    CST_YUV_420_NV21_TO_BGRA        = 97,
    CST_YUV_420_SP_TO_RGBA          = 96,
    CST_YUV_420_SP_TO_BGRA          = 97,
    CST_YUV_420_YV12_TO_RGB         = 98,
    CST_YUV_420_YV12_TO_BGR         = 99,
    CST_YUV_420_IYUV_TO_RGB         = 100,
    CST_YUV_420_IYUV_TO_BGR         = 101,
    CST_YUV_420_I420_TO_RGB         = 100,
    CST_YUV_420_I420_TO_BGR         = 101,
    CST_YUV_420_P_TO_RGB            = 98,
    CST_YUV_420_P_TO_BGR            = 99,
    CST_YUV_420_YV12_TO_RGBA        = 102,
    CST_YUV_420_YV12_TO_BGRA        = 103,
    CST_YUV_420_IYUV_TO_RGBA        = 104,
    CST_YUV_420_IYUV_TO_BGRA        = 105,
    CST_YUV_420_I420_TO_RGBA        = 104,
    CST_YUV_420_I420_TO_BGRA        = 105,
    CST_YUV_420_P_TO_RGBA           = 102,
    CST_YUV_420_P_TO_BGRA           = 103,
    CST_YUV_420_TO_GRAY             = 106,
    CST_YUV_420_NV21_TO_GRAY        = 106,
    CST_YUV_420_NV12_TO_GRAY        = 106,
    CST_YUV_420_YV12_TO_GRAY        = 106,
    CST_YUV_420_IYUV_TO_GRAY        = 106,
    CST_YUV_420_I420_TO_GRAY        = 106,
```

```
    CST_YUV_420_SP_TO_GRAY            = 106,
    CST_YUV_420_P_TO_GRAY             = 106,
    CST_YUV_422_UYVY_TO_RGB           = 107,
    CST_YUV_422_UYVY_TO_BGR           = 108,
    CST_YUV_422_Y422_TO_RGB           = 107,
    CST_YUV_422_Y422_TO_BGR           = 108,
    CST_YUV_422_UYNV_TO_RGB           = 107,
    CST_YUV_422_UYNV_TO_BGR           = 108,
    CST_YUV_422_UYVY_TO_RGBA          = 111,
    CST_YUV_422_UYVY_TO_BGRA          = 112,
    CST_YUV_422_Y422_TO_RGBA          = 111,
    CST_YUV_422_Y422_TO_BGRA          = 112,
    CST_YUV_422_UYNV_TO_RGBA          = 111,
    CST_YUV_422_UYNV_TO_BGRA          = 112,
    CST_YUV_422_YUY2_TO_RGB           = 115,
    CST_YUV_422_YUY2_TO_BGR           = 116,
    CST_YUV_422_YVYU_TO_RGB           = 117,
    CST_YUV_422_YVYU_TO_BGR           = 118,
    CST_YUV_422_YUYV_TO_RGB           = 115,
    CST_YUV_422_YUYV_TO_BGR           = 116,
    CST_YUV_422_YUNV_TO_RGB           = 115,
    CST_YUV_422_YUNV_TO_BGR           = 116,
    CST_YUV_422_YUY2_TO_RGBA          = 119,
    CST_YUV_422_YUY2_TO_BGRA          = 120,
    CST_YUV_422_YVYU_TO_RGBA          = 121,
    CST_YUV_422_YVYU_TO_BGRA          = 122,
    CST_YUV_422_YUYV_TO_RGBA          = 119,
    CST_YUV_422_YUYV_TO_BGRA          = 120,
    CST_YUV_422_YUNV_TO_RGBA          = 119,
    CST_YUV_422_YUNV_TO_BGRA          = 120,
    CST_YUV_422_UYVY_TO_GRAY          = 123,
    CST_YUV_422_YUY2_TO_GRAY          = 124,
    CST_YUV_422_Y422_TO_GRAY          = 123,
    CST_YUV_422_UYNV_TO_GRAY          = 123,
    CST_YUV_422_YVYU_TO_GRAY          = 124,
    CST_YUV_422_YUYV_TO_GRAY          = 124,
    CST_YUV_422_YUNV_TO_GRAY          = 124,
    CST_RGBA_TO_PREMULTIPLICATED_RGBA = 125,
    CST_PREMULTIPLICATED_RGBA_TO_RGBA = 126,
    CST_RGB_TO_YUV_420_I420           = 127,
    CST_BGR_TO_YUV_420_I420           = 128,
    CST_RGB_TO_YUV_420_IYUV           = 127,
    CST_BGR_TO_YUV_420_IYUV           = 128,
    CST_RGBA_TO_YUV_420_I420          = 129,
    CST_BGRA_TO_YUV_420_I420          = 130,
    CST_RGBA_TO_YUV_420_IYUV          = 129,
    CST_BGRA_TO_YUV_420_IYUV          = 130,
    CST_RGB_TO_YUV_420_YV12           = 131,
    CST_BGR_TO_YUV_420_YV12           = 132,
    CST_RGBA_TO_YUV_420_YV12          = 133,
    CST_BGRA_TO_YUV_420_YV12          = 134,
    CST_BAYER_RG_TO_BGR_EA            = 135,
    CST_BAYER_GR_TO_BGR_EA            = 136,
    CST_BAYER_BG_TO_BGR_EA            = 137,
    CST_BAYER_GB_TO_BGR_EA            = 138,
    CST_BAYER_RG_TO_RGB_EA            = 137,
    CST_BAYER_GR_TO_RGB_EA            = 138,
    CST_BAYER_BG_TO_RGB_EA            = 135,
    CST_BAYER_GB_TO_RGB_EA            = 136,
    CST_BAYER_RG_TO_BGRA              = 139,
    CST_BAYER_GR_TO_BGRA              = 140,
    CST_BAYER_BG_TO_BGRA              = 141,
    CST_BAYER_GB_TO_BGRA              = 142,
    CST_BAYER_RG_TO_RGBA              = 141,
    CST_BAYER_GR_TO_RGBA              = 142,
    CST_BAYER_BG_TO_RGBA              = 139,
    CST_BAYER_GB_TO_RGBA              = 140,
    CST_MAX                           = 143
};
```

Values

Name	Description
CST_BGR_TO_BGRA	Transform the color space from BGR to BGRA.
CST_RGB_TO_RGBA	Transform the color space from RGB to RGBA.
CST_BGRA_TO_BGR	Transform the color space from BGRA to BGR.
CST_RGBA_TO_RGB	Transform the color space from RGBA to RGB.
CST_BGR_TO_RGBA	Transform the color space from BGR to RGBA.
CST_RGB_TO_BGRA	Transform the color space from RGB to BGRA.
CST_BGRA_TO_RGB	Transform the color space from BGRA to RGB.
CST_RGBA_TO_BGR	Transform the color space from RGBA to BGR.
CST_BGR_TO_RGB	Transform the color space from BGR to RGB.
CST_RGB_TO_BGR	Transform the color space from RGB to BGR.
CST_BGRA_TO_RGBA	Transform the color space from BGRA to RGBA.
CST_RGBA_TO_BGRA	Transform the color space from RGBA to BGRA.
CST_BGR_TO_GRAY	Transform the color space from BGR to Gray.
CST_RGB_TO_GRAY	Transform the color space from RGB to Gray.
CST_GRAY_TO_BGR	Transform the color space from Gray to BGR.
CST_GRAY_TO_RGB	Transform the color space from Gray to RGB.
CST_GRAY_TO_BGRA	Transform the color space from Gray to BGRA.
CST_GRAY_TO_RGBA	Transform the color space from Gray to RGBA.
CST_BGRA_TO_GRAY	Transform the color space from BGRA to Gray.
CST_RGBA_TO_GRAY	Transform the color space from RGBA to Gray.
CST_RGB_TO_BGR_565	Transform the color space from RGB(24 bit 888) to BGR(16 bit 565).
CST_BGR_TO_BGR_565	Transform the color space from BGR(24 bit 888) to BGR(16 bit 565).
CST_BGR_565_TO_RGB	Transform the color space from BGR(16 bit 565) to RGB(24 bit 888).
CST_BGR_565_TO_BGR	Transform the color space from BGR(16 bit 565) to BGR(24 bit 888).
CST_RGBA_TO_BGR_565	Transform the color space from RGBA(32 bit 8888) to BGR(16 bit 565).
CST_BGRA_TO_BGR_565	Transform the color space from BGRA(32 bit 8888) to BGR(16 bit 565).
CST_BGR_565_TO_RGBA	Transform the color space from BGR(16 bit 565) to RGBA(32 bit 8888).
CST_BGR_565_TO_BGRA	Transform the color space from BGR(16 bit 565) to BGRA(32 bit 8888).
CST_GRAY_TO_BGR_565	Transform the color space from Gray to BGR(16 bit 565).
CST_BGR_565_TO_GRAY	Transform the color space from BGR(16 bit 565) to Gray.
CST_RGB_TO_BGR_555	Transform the color space from RGB(24 bit 888) to BGR(16 bit 555).
CST_BGR_TO_BGR_555	Transform the color space from BGR(24 bit 888) to BGR(16 bit 555).
CST_BGR_555_TO_RGB	Transform the color space from BGR(16 bit 555) to RGB(24 bit 888).
CST_BGR_555_TO_BGR	Transform the color space from BGR(16 bit 555) to BGR(24 bit 888).
CST_RGBA_TO_BGR_555	Transform the color space from RGBA(32 bit 8888) to BGR(16 bit 555).
CST_BGRA_TO_BGR_555	Transform the color space from BGRA(32 bit 8888) to BGR(16 bit 555).

Name	Description
CST_BGR_555_TO_RGBA	Transform the color space from BGR(16 bit 555) to RGBA(32 bit 8888).
CST_BGR_555_TO_BGRA	Transform the color space from BGR(16 bit 555) to BGRA(32 bit 8888).
CST_GRAY_TO_BGR_555	Transform the color space from Gray to BGR(16 bit 555).
CST_BGR_555_TO_GRAY	Transform the color space from BGR(16 bit 555) to Gray.
CST_BGR_TO_XYZ	Transform the color space from BGR to CIE XYZ (scaled to the full value range of the image).
CST_RGB_TO_XYZ	Transform the color space from RGB to CIE XYZ (scaled to the full value range of the image).
CST_XYZ_TO_BGR	Transform the color space from CIE XYZ (scaled to the full value range of the image) to BGR.
CST_XYZ_TO_RGB	Transform the color space from CIE XYZ (scaled to the full value range of the image) to RGB.
CST_BGR_TO_YCRCB	Transform the color space from BGR to YCrCb (scaled to the full value range of the image).
CST_RGB_TO_YCRCB	Transform the color space from RGB to YCrCb (scaled to the full value range of the image).
CST_YCRCB_TO_BGR	Transform the color space from YCrCb (scaled to the full value range of the image) to BGR.
CST_YCRCB_TO_RGB	Transform the color space from YCrCb (scaled to the full value range of the image) to RGB.
CST_BGR_TO_HSV	Transform the color space from BGR to HSV (for images of type USINT, H is scaled to a range from 0 to 180 and S,V from 0 to 255).
CST_RGB_TO_HSV	Transform the color space from RGB to HSV (for images of type USINT, H is scaled to a range from 0 to 180 and S,V from 0 to 255).
CST_BGR_TO_LAB	Transform the color space from BGR to CIE L*a*b* (for images of type USINT, all channels are scaled to a range from 0 to 255).
CST_RGB_TO_LAB	Transform the color space from RGB to CIE L*a*b* (for images of type USINT, all channels are scaled to a range from 0 to 255).
CST_BAYER_RG_TO_BGR	Transform the color space from BayerRG to BGR.
CST_BAYER_BG_TO_RGB	Transform the color space from BayerBG to RGB.
CST_BAYER_GR_TO_BGR	Transform the color space from BayerGR to BGR.
CST_BAYER_GB_TO_RGB	Transform the color space from BayerGB to RGB.
CST_BAYER_BG_TO_BGR	Transform the color space from BayerBG to BGR.
CST_BAYER_RG_TO_RGB	Transform the color space from BayerRG to RGB.
CST_BAYER_GB_TO_BGR	Transform the color space from BayerGB to BGR.
CST_BAYER_GR_TO_RGB	Transform the color space from BayerGR to RGB.
CST_BGR_TO_LUV	Transform the color space from BGR to CIE L*u*v* (for images of type USINT, all channels are scaled to a range from 0 to 255).
CST_RGB_TO_LUV	Transform the color space from RGB to CIE L*u*v* (for images of type USINT, all channels are scaled to a range from 0 to 255).
CST_BGR_TO_HLS	Transform the color space from BGR to HLS (for images of type USINT, H is scaled to a range from 0 to 180 and L,S from 0 to 255).

Name	Description
CST_RGB_TO_HLS	Transform the color space from RGB to HLS (for images of type USINT, H is scaled to a range from 0 to 180 and L,S from 0 to 255).
CST_HSV_TO_BGR	Transform the color space from HSV (for images of type USINT, H is scaled to a range from 0 to 180 and S,V from 0 to 255) to BGR.
CST_HSV_TO_RGB	Transform the color space from HSV (for images of type USINT, H is scaled to a range from 0 to 180 and S,V from 0 to 255) to RGB.
CST_LAB_TO_BGR	Transform the color space from CIE L*a*b* (for images of type USINT, all channels are scaled to a range from 0 to 255) to BGR.
CST_LAB_TO_RGB	Transform the color space from CIE L*a*b* (for images of type USINT, all channels are scaled to a range from 0 to 255) to RGB.
CST_LUV_TO_BGR	Transform the color space from CIE L*u*v* (for images of type USINT, all channels are scaled to a range from 0 to 255) to BGR.
CST_LUV_TO_RGB	Transform the color space from CIE L*u*v* (for images of type USINT, all channels are scaled to a range from 0 to 255) to RGB.
CST_HLS_TO_BGR	Transform the color space from HLS (for images of type USINT, H is scaled to a range from 0 to 180 and L,S from 0 to 255) to BGR.
CST_HLS_TO_RGB	Transform the color space from HLS (for images of type USINT, H is scaled to a range from 0 to 180 and L,S from 0 to 255) to RGB.
CST_BAYER_RG_TO_BGR_VNG	Transform the color space from BayerRG to BGR, using Variable Number of Gradients.
CST_BAYER_GR_TO_BGR_VNG	Transform the color space from BayerGR to BGR, using Variable Number of Gradients.
CST_BAYER_BG_TO_BGR_VNG	Transform the color space from BayerBG to BGR, using Variable Number of Gradients.
CST_BAYER_GB_TO_BGR_VNG	Transform the color space from BayerGB to BGR, using Variable Number of Gradients.
CST_BAYER_RG_TO_RGB_VNG	Transform the color space from BayerRG to RGB, using Variable Number of Gradients.
CST_BAYER_GR_TO_RGB_VNG	Transform the color space from BayerGR to RGB, using Variable Number of Gradients.
CST_BAYER_BG_TO_RGB_VNG	Transform the color space from BayerBG to RGB, using Variable Number of Gradients.
CST_BAYER_GB_TO_RGB_VNG	Transform the color space from BayerGB to RGB, using Variable Number of Gradients.
CST_BGR_TO_HSV_FULL	Transform the color space from BGR to HSV (for images of type USINT, all channels are scaled to a range from 0 to 255).
CST_RGB_TO_HSV_FULL	Transform the color space from RGB to HSV (for images of type USINT, all channels are scaled to a range from 0 to 255).
CST_BGR_TO_HLS_FULL	Transform the color space from BGR to HLS (for images of type USINT, all channels are scaled to a range from 0 to 255).
CST_RGB_TO_HLS_FULL	Transform the color space from RGB to HLS (for images of type USINT, all channels are scaled to a range from 0 to 255).

Name	Description
CST_HSV_TO_BGR_FULL	Transform the color space from HSV (for images of type USINT, all channels are scaled to a range from 0 to 255) to BGR.
CST_HSV_TO_RGB_FULL	Transform the color space from HSV (for images of type USINT, all channels are scaled to a range from 0 to 255) to RGB.
CST_HLS_TO_BGR_FULL	Transform the color space from HLS (for images of type USINT, all channels are scaled to a range from 0 to 255) to BGR.
CST_HLS_TO_RGB_FULL	Transform the color space from HLS (for images of type USINT, all channels are scaled to a range from 0 to 255) to RGB.
CST_LBGR_TO_LAB	Transform the color space from LBGR to CIE L*a*b* (for images of type USINT, all channels are scaled to a range from 0 to 255).
CST_LRGB_TO_LAB	Transform the color space from LRGB to CIE L*a*b* (for images of type USINT, all channels are scaled to a range from 0 to 255).
CST_LBGR_TO_LUV	Transform the color space from LBGR to CIE L*u*v* (for images of type USINT, all channels are scaled to a range from 0 to 255).
CST_LRGB_TO_LUV	Transform the color space from LRGB to CIE L*u*v* (for images of type USINT, all channels are scaled to a range from 0 to 255).
CST_LAB_TO_LBGR	Transform the color space from CIE L*a*b* (for images of type USINT, all channels are scaled to a range from 0 to 255) to LBGR.
CST_LAB_TO_LRGB	Transform the color space from CIE L*a*b* (for images of type USINT, all channels are scaled to a range from 0 to 255) to LRGB.
CST_LUV_TO_LBGR	Transform the color space from CIE L*u*v* (for images of type USINT, all channels are scaled to a range from 0 to 255) to LBGR.
CST_LUV_TO_LRGB	Transform the color space from CIE L*u*v* (for images of type USINT, all channels are scaled to a range from 0 to 255) to LRGB.
CST_BGR_TO_YUV	Transform the color space from BGR to YUV (scaled to the full value range of the image).
CST_RGB_TO_YUV	Transform the color space from RGB to YUV (scaled to the full value range of the image).
CST_YUV_TO_BGR	Transform the color space from YUV (scaled to the full value range of the image) to BGR.
CST_YUV_TO_RGB	Transform the color space from YUV (scaled to the full value range of the image) to RGB.
CST_BAYER_RG_TO_GRAY	Transform the color space from BayerRG to Gray.
CST_BAYER_GR_TO_GRAY	Transform the color space from BayerGR to Gray.
CST_BAYER_BG_TO_GRAY	Transform the color space from BayerBG to Gray.
CST_BAYER_GB_TO_GRAY	Transform the color space from BayerGB to Gray.
CST_YUV_420_NV12_TO_RGB	Transform the color space from YUV420 NV12 to RGB.
CST_YUV_420_NV12_TO_BGR	Transform the color space from YUV420 NV12 to BGR.
CST_YUV_420_NV21_TO_RGB	Transform the color space from YUV420 NV21 (SP) to RGB.

Name	Description
CST_YUV_420_NV21_TO_BGR	Transform the color space from YUV420 NV21 (SP) to BGR.
CST_YUV_420_SP_TO_RGB	Transform the color space from YUV420 NV21 (SP) to RGB.
CST_YUV_420_SP_TO_BGR	Transform the color space from YUV420 NV21 (SP) to BGR.
CST_YUV_420_NV12_TO_RGBA	Transform the color space from YUV420 NV12 to RGBA.
CST_YUV_420_NV12_TO_BGRA	Transform the color space from YUV420 NV12 to BGRA.
CST_YUV_420_NV21_TO_RGBA	Transform the color space from YUV420 NV21 (SP) to RGBA.
CST_YUV_420_NV21_TO_BGRA	Transform the color space from YUV420 NV21 (SP) to BGRA.
CST_YUV_420_SP_TO_RGBA	Transform the color space from YUV420 NV21 (SP) to RGBA.
CST_YUV_420_SP_TO_BGRA	Transform the color space from YUV420 NV21 (SP) to BGRA.
CST_YUV_420_YV12_TO_RGB	Transform the color space from YUV420 YV12 (P) to RGB.
CST_YUV_420_YV12_TO_BGR	Transform the color space from YUV420 YV12 (P) to BGR.
CST_YUV_420_IYUV_TO_RGB	Transform the color space from YUV420 IYUV (I420) to RGB.
CST_YUV_420_IYUV_TO_BGR	Transform the color space from YUV420 IYUV (I420) to BGR.
CST_YUV_420_I420_TO_RGB	Transform the color space from YUV420 IYUV (I420) to RGB.
CST_YUV_420_I420_TO_BGR	Transform the color space from YUV420 IYUV (I420) to BGR.
CST_YUV_420_P_TO_RGB	Transform the color space from YUV420 YV12 (P) to RGB.
CST_YUV_420_P_TO_BGR	Transform the color space from YUV420 YV12 (P) to BGR.
CST_YUV_420_YV12_TO_RGBA	Transform the color space from YUV420 YV12 (P) to RGBA.
CST_YUV_420_YV12_TO_BGRA	Transform the color space from YUV420 YV12 (P) to BGRA.
CST_YUV_420_IYUV_TO_RGBA	Transform the color space from YUV420 IYUV (I420) to RGBA.
CST_YUV_420_IYUV_TO_BGRA	Transform the color space from YUV420 IYUV (I420) to BGRA.
CST_YUV_420_I420_TO_RGBA	Transform the color space from YUV420 IYUV (I420) to RGBA.
CST_YUV_420_I420_TO_BGRA	Transform the color space from YUV420 IYUV (I420) to BGRA.
CST_YUV_420_P_TO_RGBA	Transform the color space from YUV420 YV12 (P) to RGBA.
CST_YUV_420_P_TO_BGRA	Transform the color space from YUV420 YV12 (P) to BGRA.
CST_YUV_420_TO_GRAY	Transform the color space from YUV420 to Gray.
CST_YUV_420_NV21_TO_GRAY	Transform the color space from YUV420 to Gray.
CST_YUV_420_NV12_TO_GRAY	Transform the color space from YUV420 to Gray.
CST_YUV_420_YV12_TO_GRAY	Transform the color space from YUV420 to Gray.

Name	Description
CST_YUV_420_IYUV_TO_GRAY	Transform the color space from YUV420 to Gray.
CST_YUV_420_I420_TO_GRAY	Transform the color space from YUV420 to Gray.
CST_YUV_420_SP_TO_GRAY	Transform the color space from YUV420 to Gray.
CST_YUV_420_P_TO_GRAY	Transform the color space from YUV420 to Gray.
CST_YUV_422_UYVY_TO_RGB	Transform the color space from YUV422 UYVY (Y422, UYNV) to RGB.
CST_YUV_422_UYVY_TO_BGR	Transform the color space from YUV422 UYVY (Y422, UYNV) to BGR.
CST_YUV_422_Y422_TO_RGB	Transform the color space from YUV422 UYVY (Y422, UYNV) to RGB.
CST_YUV_422_Y422_TO_BGR	Transform the color space from YUV422 UYVY (Y422, UYNV) to BGR.
CST_YUV_422_UYNV_TO_RGB	Transform the color space from YUV422 UYVY (Y422, UYNV) to RGB.
CST_YUV_422_UYNV_TO_BGR	Transform the color space from YUV422 UYVY (Y422, UYNV) to BGR.
CST_YUV_422_UYVY_TO_RGBA	Transform the color space from YUV422 UYVY (Y422, UYNV) to RGBA.
CST_YUV_422_UYVY_TO_BGRA	Transform the color space from YUV422 UYVY (Y422, UYNV) to BGRA.
CST_YUV_422_Y422_TO_RGBA	Transform the color space from YUV422 UYVY (Y422, UYNV) to RGB.
CST_YUV_422_Y422_TO_BGRA	Transform the color space from YUV422 UYVY (Y422, UYNV) to BGRA.
CST_YUV_422_UYNV_TO_RGBA	Transform the color space from YUV422 UYVY (Y422, UYNV) to RGB.
CST_YUV_422_UYNV_TO_BGRA	Transform the color space from YUV422 UYVY (Y422, UYNV) to BGRA.
CST_YUV_422_YUY2_TO_RGB	Transform the color space from YUV422 YUYV (YUY2, YUNV) to RGB.
CST_YUV_422_YUY2_TO_BGR	Transform the color space from YUV422 YUYV (YUY2, YUNV) to BGR.
CST_YUV_422_YVYU_TO_RGB	Transform the color space from YUV422 YVYU to RGB.
CST_YUV_422_YVYU_TO_BGR	Transform the color space from YUV422 YVYU to BGR.
CST_YUV_422_YUYV_TO_RGB	Transform the color space from YUV422 YUYV (YUY2, YUNV) to RGB.
CST_YUV_422_YUYV_TO_BGR	Transform the color space from YUV422 YUYV (YUY2, YUNV) to BGR.
CST_YUV_422_YUNV_TO_RGB	Transform the color space from YUV422 YUYV (YUY2, YUNV) to RGB.
CST_YUV_422_YUNV_TO_BGR	Transform the color space from YUV422 YUYV (YUY2, YUNV) to BGR.
CST_YUV_422_YUY2_TO_RGBA	Transform the color space from YUV422 YUYV (YUY2, YUNV) to RGBA.
CST_YUV_422_YUY2_TO_BGRA	Transform the color space from YUV422 YUYV (YUY2, YUNV) to BGRA.
CST_YUV_422_YVYU_TO_RGBA	Transform the color space from YUV422 YVYU to RGBA.
CST_YUV_422_YVYU_TO_BGRA	Transform the color space from YUV422 YVYU to BGRA.
CST_YUV_422_YUYV_TO_RGBA	Transform the color space from YUV422 YUYV (YUY2, YUNV) to RGBA.

Name	Description
CST_YUV_422_YUYV_TO_BGRA	Transform the color space from YUV422 YUYV (YUY2, YUNV) to BGRA.
CST_YUV_422_YUNV_TO_RGBA	Transform the color space from YUV422 YUYV (YUY2, YUNV) to RGBA.
CST_YUV_422_YUNV_TO_BGRA	Transform the color space from YUV422 YUYV (YUY2, YUNV) to BGRA.
CST_YUV_422_UYVY_TO_GRAY	Transform the color space from YUV422 UYVY (Y422, UYNV) to Gray.
CST_YUV_422_YUY2_TO_GRAY	Transform the color space from YUV422 YUYV (YUY2, YUNV) to Gray.
CST_YUV_422_Y422_TO_GRAY	Transform the color space from YUV422 UYVY (Y422, UYNV) to Gray.
CST_YUV_422_UYNV_TO_GRAY	Transform the color space from YUV422 UYVY (Y422, UYNV) to Gray.
CST_YUV_422_YVYU_TO_GRAY	Transform the color space from YUV422 YVYU to Gray.
CST_YUV_422_YUYV_TO_GRAY	Transform the color space from YUV422 YUYV (YUY2, YUNV) to Gray.
CST_YUV_422_YUNV_TO_GRAY	Transform the color space from YUV422 YUYV (YUY2, YUNV) to Gray.
CST_RGBA_TO_PREMULTIPLICATED_RGBA	Transform the color space from RGBA to premultiplicated RGBA.
CST_PREMULTIPLICATED_RGBA_TO_RGBA	Transform the color space from premultiplicated RGBA to RGBA.
CST_RGB_TO_YUV_420_I420	Transform the color space from RGB to YUV420 IYUV (I420).
CST_BGR_TO_YUV_420_I420	Transform the color space from BGR to YUV420 IYUV (I420).
CST_RGB_TO_YUV_420_IYUV	Transform the color space from RGB to YUV420 IYUV (I420).
CST_BGR_TO_YUV_420_IYUV	Transform the color space from BGR to YUV420 IYUV (I420).
CST_RGBA_TO_YUV_420_I420	Transform the color space from RGBA to YUV420 IYUV (I420).
CST_BGRA_TO_YUV_420_I420	Transform the color space from BGRA to YUV420 IYUV (I420).
CST_RGBA_TO_YUV_420_IYUV	Transform the color space from RGBA to YUV420 IYUV (I420).
CST_BGRA_TO_YUV_420_IYUV	Transform the color space from BGRA to YUV420 IYUV (I420).
CST_RGB_TO_YUV_420_YV12	Transform the color space from RGB to YUV420 YV12 (P).
CST_BGR_TO_YUV_420_YV12	Transform the color space from BGR to YUV420 YV12 (P).
CST_RGBA_TO_YUV_420_YV12	Transform the color space from RGBA to YUV420 YV12 (P).
CST_BGRA_TO_YUV_420_YV12	Transform the color space from BGRA to YUV420 YV12 (P).
CST_BAYER_RG_TO_BGR_EA	Transform the color space from BayerRG to BGR, using an Edge Aware algorithm.
CST_BAYER_GR_TO_BGR_EA	Transform the color space from BayerGR to BGR, using an Edge Aware algorithm.
CST_BAYER_BG_TO_BGR_EA	Transform the color space from BayerBG to BGR, using an Edge Aware algorithm.

Name	Description
CST_BAYER_GB_TO_BGR_EA	Transform the color space from BayerGB to BGR, using an Edge Aware algorithm.
CST_BAYER_RG_TO_RGB_EA	Transform the color space from BayerRG to RGB, using an Edge Aware algorithm.
CST_BAYER_GR_TO_RGB_EA	Transform the color space from BayerGR to RGB, using an Edge Aware algorithm.
CST_BAYER_BG_TO_RGB_EA	Transform the color space from BayerBG to RGB, using an Edge Aware algorithm.
CST_BAYER_GB_TO_RGB_EA	Transform the color space from BayerGB to RGB, using an Edge Aware algorithm.
CST_BAYER_RG_TO_BGRA	Transform the color space from BayerRG to BGRA.
CST_BAYER_GR_TO_BGRA	Transform the color space from BayerGR to BGRA.
CST_BAYER_BG_TO_BGRA	Transform the color space from BayerBG to BGRA.
CST_BAYER_GB_TO_BGRA	Transform the color space from BayerGB to BGRA.
CST_BAYER_RG_TO_RGBA	Transform the color space from BayerRG to RGBA.
CST_BAYER_GR_TO_RGBA	Transform the color space from BayerGR to RGBA.
CST_BAYER_BG_TO_RGBA	Transform the color space from BayerBG to RGBA.
CST_BAYER_GB_TO_RGBA	Transform the color space from BayerGB to RGBA.
CST_MAX	For internal use only, adapted when adding new values.

6.2.1.4.15 ETcVnColorTrainingMethod

Offers color training methods.

Syntax

Definition:

```
enum ETcVnColorTrainingMethod : LONG
{
    CTM_LAB = 0,
    CTM_RGB = 1
};
```

Values

Name	Description
CTM_LAB	Trains the color in CIE L*a*b* color space.
CTM_RGB	Trains the color in RGB color space.

6.2.1.4.16 ETcVnConnectedComponentsAlgorithm

Offers connected components algorithms.

Syntax

Definition:

```
enum ETcVnConnectedComponentsAlgorithm : LONG
{
    CCA_WU    = 0,
    CCA_GRANA = 1
};
```

Values

Name	Description
CCA_WU	SAUF algorithm
CCA_GRANA	BBDT algorithm for 8-way connectivity, SAUF algorithm for 4-way connectivity

6.2.1.4.17 ETcVnContainerExportFormat

Offers container export formats.

Syntax

Definition:

```
enum ETcVnContainerExportFormat : USHORT
{
    CEF_XML            = 0,
    CEF_XML_SERIALIZED = 1,
    CEF_CSV            = 2,
    CEF_BINARY         = 3
};
```

Values

Name	Description
CEF_XML	XML format with human readable data
CEF_XML_SERIALIZED	XML format with serialized data
CEF_CSV	CSV format (limited to two dimensional representation, so not applicable for all container types)
CEF_BINARY	Binary serialized data (recommended for large containers)

6.2.1.4.18 ETcVnContourApproximationMethod

Offers methods for contour approximation.

Syntax

Definition:

```
enum ETcVnContourApproximationMethod : LONG
{
    CAM_NONE      = 1,
    CAM_SIMPLE    = 2,
    CAM_TC89_L1   = 3,
    CAM_TC89_KCOS = 4
};
```

Values

Name	Description
CAM_NONE	No approximation, every single point is stored, i.e. two subsequent points are always direct neighbors (horizontal, vertical or diagonal).
CAM_SIMPLE	Lossless compression of segments that form straight lines in horizontal, vertical or diagonal direction leaving only their endpoints (For instance, an up-right rectangle is reduced to its four corner points.)
CAM_TC89_L1	Compression using a Teh-Chin chain approximation algorithm (IEEE-paper C. Teh, R. Chin, On the Detection of Dominant Points on Digital Curves, 1989).
CAM_TC89_KCOS	Compression using a Teh-Chin chain approximation algorithm (IEEE-paper C. Teh, R. Chin, On the Detection of Dominant Points on Digital Curves, 1989).

6.2.1.4.19 ETcVnContourRetrievalMode

Offers retrieval modes for a contour search.

Syntax

Definition:

```
enum ETcVnContourRetrievalMode : LONG
{
    CRM_EXTERNAL             = 0,
    CRM_LIST                 = 1,
    CRM_CONNECTED_COMPONENTS = 2,
    CRM_TREE                 = 3,
    CRM_FLOODFILL            = 4
};
```

Values

Name	Description
CRM_EXTERNAL	Returns only external contours.
CRM_LIST	Returns all found contours, not considering their hierarchy.
CRM_CONNECTED_COMPONENTS	Returns all contours with a 2-level hierarchy (External contours are assigned level 0, internal contours level 1).
CRM_TREE	Returns all contours and their full hierarchy.
CRM_FLOODFILL	Returns the found contours using a floodfill algorithm (only available for DINT images).

6.2.1.4.20 ETcVnContoursMatchComparisonMethod

Offers comparison methods for contour matching.

Syntax

Definition:

```
enum ETcVnContoursMatchComparisonMethod : LONG
{
    CMCM_CONTOURS_MATCH_I1 = 1,
    CMCM_CONTOURS_MATCH_I2 = 2,
    CMCM_CONTOURS_MATCH_I3 = 3
};
```

Values

Name	Description
CMCM_CONTOURS_MATCH_I1	Contours are compared using the sum over the differences between the reciprocal individual characteristics.
CMCM_CONTOURS_MATCH_I2	Contours are compared using the sum over the differences between the individual characteristics.
CMCM_CONTOURS_MATCH_I3	Contours are compared using only the maximum difference between the individual characteristics.

6.2.1.4.21 ETcVnDiffusivityTypeKAZE

Offers diffusivity types for feature detection methods KAZE and AKAZE.

Syntax

Definition:

```
enum ETcVnDiffusivityTypeKAZE : LONG
{
    DT1_KAZE_PM_G1       = 0,
    DT1_KAZE_PM_G2       = 1,
    DT1_KAZE_WEICKERT    = 2,
    DT1_KAZE_CHARBONNIER = 3
};
```

Values

Name	Description
DT1_KAZE_PM_G1	Uses the Perona-Malik diffusivity filter g1.
DT1_KAZE_PM_G2	Uses the Perona-Malik diffusivity filter g2.
DT1_KAZE_WEICKERT	Uses the Weickert diffusivity filter.
DT1_KAZE_CHARBONNIER	Uses the Charbonnier diffusivity filter.

6.2.1.4.22 ETcVnDistanceTransformationLabel

Offers types of the label array to build.

Syntax

Definition:

```
enum ETcVnDistanceTransformationLabel : LONG
{
    DTL_CCOMP = 0,
    DTL_PIXEL = 1
};
```

Values

Name	Description
DTL_CCOMP	Labels connected components (Each connected component of zeros in the source image and the pixels closest to the connected component will be assigned the same label).
DTL_PIXEL	Labels pixels (Each zero pixel and the non-zero pixels closest to it get their own label).

6.2.1.4.23 ETcVnDistanceTransformationMask

Offers different sizes of the distance transformation mask. Please note, that some sizes are not supported by certain distance types.

Syntax

Definition:

```
enum ETcVnDistanceTransformationMask : LONG
{
    DTM_PRECISE = 0,
    DTM_3       = 3,
    DTM_5       = 5
};
```

Values

Name	Description
DTM_PRECISE	Uses a precise mask.
DTM_3	Uses a 3x3 mask.
DTM_5	Uses a 5x5 mask.

6.2.1.4.24 ETcVnDistanceType

Offers distance types.

Syntax

Definition:

```
enum ETcVnDistanceType : LONG
{
    DT_USER   = -1,
    DT_L1     = 1,
    DT_L2     = 2,
    DT_C      = 3,
    DT_L12    = 4,
    DT_FAIR   = 5,
    DT_WELSCH = 6,
    DT_HUBER  = 7
};
```

Values

Name	Description
DT_USER	User defined distance
DT_L1	|x1 - x2| + |y1 - y2|
DT_L2	Euclidean distance
DT_C	max(|x1 - x2|, |y1 - y2|)
DT_L12	L1 - L2 : 2*(sqrt(1 + x*x/2) - 1))
DT_FAIR	c^2(|x|/c-log(1+|x|/c)), c = 1.3998
DT_WELSCH	c^2/2(1-exp(-(x/c)^2)), c = 2.9846
DT_HUBER	|x|<c ? x^2/2 : c(|x|-c/2), c=1.345

6.2.1.4.25 ETcVnDrawMatchesFlags

Offers a combination of flags to support overdrawing an existing destination image and/or skipping single (i.e. non-matched) keypoints and/or drawing additional (rich-)keypoint information (size and orientation). Used by F_VN_DrawKeypointsExp.

Syntax

Definition:

```
enum ETcVnDrawMatchesFlags : LONG
{
    DMF_DEFAULT                          = 0,
    DMF_OVERDRAW                         = 1,
    DMF_SKIPSINGLE                       = 2,
    DMF_OVERDRAW_SKIPSINGLE              = 3,
    DMF_RICHKEYPOINT                     = 4,
    DMF_OVERDRAW_RICHKEYPOINT            = 5,
    DMF_SKIPSINGLE_RICHKEYPOINT          = 6,
    DMF_OVERDRAW_SKIPSINGLE_RICHKEYPOINT = 7
};
```

Values

Name	Description
DMF_DEFAULT	Draw all keypoints into a new image.
DMF_OVERDRAW	Draw all keypoints into the existing destination image.
DMF_SKIPSINGLE	Draw the keypoints into a new image but skip single keypoints.
DMF_OVERDRAW_SKIPSINGLE	Draw the keypoints into the existing destination image but skip single keypoints.
DMF_RICHKEYPOINT	Draw all keypoints with rich information into a new image.
DMF_OVERDRAW_RICHKEYPOINT	Draw all keypoints with rich information into the existing destination image.
DMF_SKIPSINGLE_RICHKEYPOINT	Draw the keypoints with rich information into a new image but skip single keypoints.
DMF_OVERDRAW_SKIPSINGLE_RICHKEYPOINT	Draw the keypoints with rich information into the existing destination image but skip single keypoints.

6.2.1.4.26 ETcVnDrawShape

Offers shapes to be drawn.

Syntax

Definition:

```
enum ETcVnDrawShape : LONG
{
    DS_RANDOM  = -1,
    DS_CIRCLE  = 0,
    DS_SQUARE  = 1,
    DS_PLUS    = 2,
    DS_X       = 3,
    DS_DIAMOND = 4,
    DS_MAX     = 5
};
```

Values

Name	Description
DS_RANDOM	Randomly select a circle, square, plus, X or diamond.
DS_CIRCLE	Circle
DS_SQUARE	Square
DS_PLUS	+
DS_X	x
DS_DIAMOND	Diamond
DS_MAX	For internal use only, adapted when adding new values.

6.2.1.4.27 ETcVnEdgeDetectionAlgorithm

Offers edge detection algorithms.

Syntax

Definition:

```
enum ETcVnEdgeDetectionAlgorithm : LONG
{
    EDA_INTERPOLATION   = 0,
    EDA_APPROX_ERF      = 1,
    EDA_APPROX_GAUSSIAN = 2
};
```

Values

Name	Description
EDA_INTERPOLATION	Interpolates pixels (bilinear) and then finds the maximum gradient. This approach is fast and stable, but usually less precise than the function approximation methods.
EDA_APPROX_ERF	Approximates the edge with an erf function. This approach is slower than the interpolation method, but usually more precise. However, it can be inaccurate if the edge does not suit the erf model.
EDA_APPROX_GAUSSIAN	Approximates the edge with a gaussian function. This method is intended to find the center of relatively thin lines, so it is likely to be inaccurate for other edges.

6.2.1.4.28 ETcVnEdgeDirection

Offers edge directions relative to the search direction.

Syntax

Definition:

```
enum ETcVnEdgeDirection : LONG
{
    ED_DARK_TO_LIGHT = 0,
    ED_LIGHT_TO_DARK = 1
};
```

Values

Name	Description
ED_DARK_TO_LIGHT	Dark to light
ED_LIGHT_TO_DARK	Light to dark

6.2.1.4.29 ETcVnElementType

Offers element types.

Syntax

Definition:

```
enum ETcVnElementType : LONG
{
    ET_SAME_AS_SOURCE = -1,
    ET_USINT          = 0,
    ET_SINT           = 1,
    ET_UINT           = 2,
    ET_INT            = 3,
    ET_DINT           = 4,
```

```
    ET_REAL           = 5,
    ET_LREAL          = 6
};
```

Values

Name	Description
ET_SAME_AS_SOURCE	Sets the element type of the destination image to the source image element type.
ET_USINT	USINT (depth: 8 bit, 0..255)
ET_SINT	SINT (depth: 8 bit, -128..127)
ET_UINT	UINT (depth: 16 bit, 0..65535)
ET_INT	INT (depth: 16 bit, -32768..32767)
ET_DINT	DINT (depth: 32 bit, -2147483648..2147483647)
ET_REAL	REAL (depth: 32 bit, ~ -3.402823E 10^38 .. ~ 3.402823E 38)
ET_LREAL	LREAL (depth: 64 bit, ~ -1.79769313486231E 308 .. ~ 1.79769313486232E 308)

6.2.1.4.30 ETcVnEstimationAlgorithm

Offers estimation algorithms for matching point sets.

Syntax

Definition:

```
enum ETcVnEstimationAlgorithm : LONG
{
    EA_DEFAULT = 0,
    EA_LMEDS   = 4,
    EA_RANSAC  = 8,
    EA_RHO     = 16
};
```

Values

Name	Description
EA_DEFAULT	Use all points.
EA_LMEDS	Least Median of Squares
EA_RANSAC	Random Sample Consensus
EA_RHO	Progressive Sample Consensus

6.2.1.4.31 ETcVnExtremePointDirection

Offers search directions for the extreme point.

Syntax

Definition:

```
enum ETcVnExtremePointDirection : LONG
{
    EPD_TOP_LEFT      = 0,
    EPD_TOP_MEDIAN    = 1,
    EPD_TOP_RIGHT     = 2,
    EPD_BOTTOM_LEFT   = 3,
    EPD_BOTTOM_MEDIAN = 4,
    EPD_BOTTOM_RIGHT  = 5,
    EPD_LEFT_TOP      = 6,
    EPD_LEFT_MEDIAN   = 7,
    EPD_LEFT_BOTTOM   = 8,
    EPD_RIGHT_TOP     = 9,
```

```
    EPD_RIGHT_MEDIAN  = 10,
    EPD_RIGHT_BOTTOM  = 11
};
```

Values

Name	Description
EPD_TOP_LEFT	Find the topmost point (min y, take left one if more than 1).
EPD_TOP_MEDIAN	Find the topmost point (min y, take median one if more than 1).
EPD_TOP_RIGHT	Find the topmost point (min y, take right one if more than 1).
EPD_BOTTOM_LEFT	Find the bottommost point (max y, take left one if more than 1).
EPD_BOTTOM_MEDIAN	Find the bottommost point (max y, take median one if more than 1).
EPD_BOTTOM_RIGHT	Find the bottommost point (max y, take right one if more than 1).
EPD_LEFT_TOP	Find the leftmost point (min x, take top one if more than 1).
EPD_LEFT_MEDIAN	Find the leftmost point (min x, take median one if more than 1).
EPD_LEFT_BOTTOM	Find the leftmost point (min x, take bottom one if more than 1).
EPD_RIGHT_TOP	Find the rightmost point (max x, take top one if more than 1).
EPD_RIGHT_MEDIAN	Find the rightmost point (max x, take median one if more than 1).
EPD_RIGHT_BOTTOM	Find the rightmost point (max x, take bottom one if more than 1).

6.2.1.4.32 ETcVnFeatureDescriptorTypeAKAZE

Offers descriptor types for AKAZE method.

Syntax

Definition:

```
enum ETcVnFeatureDescriptorTypeAKAZE : LONG
{
    FDT_AKAZE_KAZE_UPRIGHT = 2,
    FDT_AKAZE_KAZE         = 3,
    FDT_AKAZE_MLDB_UPRIGHT = 4,
    FDT_AKAZE_MLDB         = 5
};
```

Values

Name	Description
FDT_AKAZE_KAZE_UPRIGHT	KAZE descriptor (upright, i.e. rotation dependend).
FDT_AKAZE_KAZE	KAZE descriptor (rotation invariant).
FDT_AKAZE_MLDB_UPRIGHT	MLDB descriptor (upright, i.e. rotation dependend).
FDT_AKAZE_MLDB	MLDB descriptor (rotation invariant).

6.2.1.4.33 ETcVnFeatureScalingType

Offers feature scaling types

Syntax

Definition:

```
enum ETcVnFeatureScalingType : LONG
{
    FST1_MAXABS          = 0,
    FST1_MINMAX          = 1,
    FST1_STANDARDIZATION = 2
};
```

Values

Name	Description
FST1_MAXABS	Use the max of the absolute values of the feature vector. The normlized values are in the range [-1 to 1] or [0 to 1] if all the original values are positive.
FST1_MINMAX	Use the min and max values of the feature vector. The normlized values are in the range [0 to 1]
FST1_STANDARDIZATION	Use the mean and the standard deviation of the feature vector. The normalized vector has a mean value equal to zero and standard deviation equal to one

6.2.1.4.34 ETcVnFeatureScoreTypeORB

Offers algorithms used to rank features.

Syntax

Definition:

```
enum ETcVnFeatureScoreTypeORB : LONG
{
    FST_ORB_HARRIS = 0,
    FST_ORB_FAST   = 1
};
```

Values

Name	Description
FST_ORB_HARRIS	Harris algorithm (produces more stable keypoints, but computation takes longer).
FST_ORB_FAST	FAST algorithm (produces slightly less stable keypoints, but computation is faster).

6.2.1.4.35 ETcVnFilterDirection

Offers directions, in which to apply filter.

Syntax

Definition:

```
enum ETcVnFilterDirection : SHORT
{
    FD_X = 0,
    FD_Y = 1
};
```

Values

Name	Description
FD_X	X-direction
FD_Y	Y-direction

6.2.1.4.36 ETcVnFlipAxis

Defines the axis around which to flip (mirror) the image.

Syntax

Definition:

```
enum ETcVnFlipAxis : SHORT
{
    FA_XY = -1,
    FA_X  = 0,
    FA_Y  = 1
};
```

Values

Name	Description
FA_XY	Flip the image around both axes
FA_X	Flip the image around the X axis
FA_Y	Flip the image around the Y axis

6.2.1.4.37 ETcVnFontType

Offers font types.

Syntax

Definition:

```
enum ETcVnFontType : LONG
{
    FT_HERSHEY_SIMPLEX              = 0,
    FT_HERSHEY_PLAIN                = 1,
    FT_HERSHEY_DUPLEX               = 2,
    FT_HERSHEY_COMPLEX              = 3,
    FT_HERSHEY_TRIPLEX              = 4,
    FT_HERSHEY_COMPLEX_SMALL        = 5,
    FT_HERSHEY_SCRIPT_SIMPLEX       = 6,
    FT_HERSHEY_SCRIPT_COMPLEX       = 7,
    FT_HERSHEY_PLAIN_ITALIC         = 17,
    FT_HERSHEY_COMPLEX_ITALIC       = 19,
    FT_HERSHEY_TRIPLEX_ITALIC       = 20,
    FT_HERSHEY_COMPLEX_SMALL_ITALIC = 21
};
```

Values

Name	Description
FT_HERSHEY_SIMPLEX	Normal size sans-serif font.
FT_HERSHEY_PLAIN	Small size sans-serif font.
FT_HERSHEY_DUPLEX	More complex normal size sans-serif font.
FT_HERSHEY_COMPLEX	Normal size serif font.
FT_HERSHEY_TRIPLEX	More complex normal size serif font.
FT_HERSHEY_COMPLEX_SMALL	Small size serif font.
FT_HERSHEY_SCRIPT_SIMPLEX	Hand-writing style font.
FT_HERSHEY_SCRIPT_COMPLEX	More complex hand-writing style font.
FT_HERSHEY_PLAIN_ITALIC	Small size sans-serif font (italic).
FT_HERSHEY_COMPLEX_ITALIC	Normal size serif font (italic).
FT_HERSHEY_TRIPLEX_ITALIC	More complex normal size serif font (italic).
FT_HERSHEY_COMPLEX_SMALL_ITALIC	Small size serif font (italic).

6.2.1.4.38 ETcVnHoughMethod

Offers Hough methods.

Syntax

Definition:

```
enum ETcVnHoughMethod : LONG
{
    HM_STANDARD      = 0,
    HM_PROBABILISTIC = 1,
    HM_MULTI_SCALE   = 2,
    HM_GRADIENT      = 3,
    HM_GRADIENT_ALT  = 4
};
```

Values

Name	Description
HM_STANDARD	Classical or standard Hough transform. Every line is represented by two floating-point numbers (f1, f2), where f1 is a distance between (0,0) origin and the line, and f2 is the angle between x-axis and the normal to the line.
HM_PROBABILISTIC	Probabilistic Hough transform. More efficient for pictures containing long linear segments. Returns line segments rather than the whole line, while each segment is represented by start and end point.
HM_MULTI_SCALE	Multi scale variant of classical Hough transform.
HM_GRADIENT	Method designed to find circles. See paper: HK Yuen, John Princen, John Illingworth, and Josef Kittler. Comparative study of hough transform methods for circle finding. Image and Vision Computing, 8(1):71–77, 1990
HM_GRADIENT_ALT	Variation of the GRADIENT method.

6.2.1.4.39 ETcVnInitializableFunction

Offers initializable functions.

Syntax

Definition:

```
enum ETcVnInitializableFunction : LONG
{
    IF_OCR = 0,
};
```

Values

Name	Description
IF_OCR	F_VN_OCR

6.2.1.4.40 ETcVnInterpolationType

Offers interpolation types.

Syntax

Definition:

```
enum ETcVnInterpolationType : LONG
{
    IT_NEAREST_NEIGHBOR = 0,
    IT_BILINEAR         = 1,
    IT_BICUBIC          = 2,
    IT_AREA_BASED       = 3,
    IT_LANCZOS4         = 4
};
```

Values

Name	Description
IT_NEAREST_NEIGHBOR	Nearest neighbor interpolation
IT_BILINEAR	Bilinear interpolation
IT_BICUBIC	Bicubic interpolation
IT_AREA_BASED	Area based interpolation
IT_LANCZOS4	Lanczos4 interpolation

6.2.1.4.41 ETcVnKeypointDetectionTypeAGAST

Offers different neighborhood types for AGAST method (For details see paper: E. Mair et al.: Adaptive and Generic Corner Detection Based on the Accelerated Segment Test, 2010).

Syntax

Definition:

```
enum ETcVnKeypointDetectionTypeAGAST : LONG
{
    KDT_AGAST_5_8   = 0,
    KDT_AGAST_7_12d = 1,
    KDT_AGAST_7_12s = 2,
    KDT_AGAST_9_16  = 3
};
```

Values

Name	Description
KDT_AGAST_5_8	5 of 8
KDT_AGAST_7_12d	7 of 12 diamond
KDT_AGAST_7_12s	7 of 12 square
KDT_AGAST_9_16	9 of 16

6.2.1.4.42 ETcVnKeypointDetectionTypeFAST

Offers different neighborhood types for FAST method (For details see paper: E. Rosten: Machine Learning for High-speed Corner Detection, 2006).

Syntax

Definition:

```
enum ETcVnKeypointDetectionTypeFAST : LONG
{
    KDT_FAST_5_8  = 0,
    KDT_FAST_7_12 = 1,
    KDT_FAST_9_16 = 2
};
```

Values

Name	Description
KDT_FAST_5_8	5 of 8
KDT_FAST_7_12	7 of 12
KDT_FAST_9_16	9 of 16

6.2.1.4.43 ETcVnKnn

Offers kNN model types.

Syntax

Definition:

```
enum ETcVnKnn : ULONG
{
    KNN_CLASSIFIER       = 1,
    KNN_NOVELTY_DETECTOR = 2,
    KNN_REGRESSOR        = 4
};
```

Values

Name	Description
KNN_CLASSIFIER	Classifier
KNN_NOVELTY_DETECTOR	Novelty detector
KNN_REGRESSOR	Regressor

6.2.1.4.44 ETcVnLineType

Offers line types.

Syntax

Definition:

```
enum ETcVnLineType : LONG
{
    LT_4_CONNECTED = 4,
    LT_8_CONNECTED = 8,
    LT_ANTIALIASED = 16
};
```

Values

Name	Description
LT_4_CONNECTED	4-connected (pixels are connected horizontally and vertically).
LT_8_CONNECTED	8-connected (pixels are connected horizontally, vertically and diagonally).
LT_ANTIALIASED	Antialiased (drawn using Gaussian filtering, only implemented for 8-bit images).

6.2.1.4.45 ETcVnMorphologicalOperator

Offers morphological operators.

Syntax

Definition:

```
enum ETcVnMorphologicalOperator : LONG
{
    MO_EROSION                      = 0x0,
    MO_DILATION                     = 0x1,
    MO_OPENING                      = 0x2,
    MO_CLOSING                      = 0x3,
    MO_GRADIENT                     = 0x4,
    MO_WHITE_TOPHAT                 = 0x5,
    MO_BLACK_TOPHAT                 = 0x6,
    MO_OPENING_BY_RECONSTRUCTION    = 0x40000002,
    MO_CLOSING_BY_RECONSTRUCTION    = 0x40000003,
    MO_WHITE_TOPHAT_BY_RECONSTRUCTION = 0x40000005,
    MO_BLACK_TOPHAT_BY_RECONSTRUCTION = 0x40000006
};
```

Values

Name	Description
MO_EROSION	Shrinks objects (regions of foreground (i.e. white) pixels), removing regions smaller than the structuring element completely.
MO_DILATION	Expands objects and thereby closes small holes inside objects.
MO_OPENING	Applies an erosion first, a dilation second. Objects smaller than the structuring element are removed while outer shapes remain largely the same.
MO_CLOSING	Applies a dilation first, an erosion second. Holes inside objects that fit into the structuring element are closed completely while outer shapes remain largely the same.
MO_GRADIENT	Difference between the dilation and the erosion of an image.
MO_WHITE_TOPHAT	Difference between an input image and its opening.
MO_BLACK_TOPHAT	Difference between an input image and its closing.
MO_OPENING_BY_RECONSTRUCTION	Opening with subsequent reconstruction of objects that were not removed by the opening.
MO_CLOSING_BY_RECONSTRUCTION	Closing with subsequent reconstruction of objects that were not removed by the closing.
MO_WHITE_TOPHAT_BY_RECONSTRUCTION	White tophat with subsequent reconstruction of objects that were not removed by the white tophat.
MO_BLACK_TOPHAT_BY_RECONSTRUCTION	Black tophat with subsequent reconstruction of objects that were not removed by the black tophat.

6.2.1.4.46 ETcVnNbc

Offers normal Bayes classifier types.

Syntax

Definition:

```
enum ETcVnNbc : ULONG
{
    NBC_CLASSIFIER       = 1,
    NBC_NOVELTY_DETECTOR = 2
};
```

Values

Name	Description
NBC_CLASSIFIER	Classifier
NBC_NOVELTY_DETECTOR	Novelty detector

6.2.1.4.47 ETcVnNormalizationType

Offers normalization types.

Syntax

Definition:

```
enum ETcVnNormalizationType : LONG
{
    NT_INF               = 1,
    NT_L1                = 2,
    NT_L2                = 4,
    NT_L2SQR             = 5,
    NT_HAMMING           = 6,
    NT_HAMMING2          = 7,
    NT_RELATIVE_INF      = 9,
    NT_RELATIVE_L1       = 10,
    NT_RELATIVE_L2       = 12,
    NT_RELATIVE_L2SQR    = 13,
    NT_RELATIVE_HAMMING  = 14,
    NT_RELATIVE_HAMMING2 = 15,
    NT_MINMAX            = 32
};
```

Values

Name	Description
NT_INF	Infinity norm
NT_L1	L1 norm
NT_L2	L2 (euclidean) norm
NT_L2SQR	Squared L2 norm
NT_HAMMING	Hamming distance (bitwise)
NT_HAMMING2	Hamming distance (each 2 bits combined to one)
NT_RELATIVE_INF	Infinity norm (when comparing 2 arrays, the norm of their difference is divided by the norm of the second array)
NT_RELATIVE_L1	L1 norm (when comparing 2 arrays, the norm of their difference is divided by the norm of the second array)
NT_RELATIVE_L2	L2 (euclidean) norm (when comparing 2 arrays, the norm of their difference is divided by the norm of the second array)
NT_RELATIVE_L2SQR	Squared L2 norm (when comparing 2 arrays, the norm of their difference is divided by the norm of the second array)
NT_RELATIVE_HAMMING	Hamming distance (bitwise; when comparing 2 arrays, the norm of their difference is divided by the norm of the second array)
NT_RELATIVE_HAMMING2	Hamming distance (each 2 bits combined to one; when comparing 2 arrays, the norm of their difference is divided by the norm of the second array)
NT_MINMAX	Normalize the values to a range given by a minimum and a maximum value.

6.2.1.4.48 ETcVnOcrModelType

Offers different OCR model types

Syntax

Definition:

```
enum ETcVnOcrModelType : ULONGLONG
{
    OMT_NUMBERS              = 580542139465730,
    OMT_NUMBERS_SC           = 580542139465732,
    OMT_UCLETTERS            = 580542139465736,
    OMT_NUMBERS_SC_UCLETTERS = 580542139465744
};
```

Values

Name	Description
OMT_NUMBERS	Classify numbers
OMT_NUMBERS_SC	Classify numbers and special characters
OMT_UCLETTERS	Classify uppercase letters
OMT_NUMBERS_SC_UCLETTERS	Classify numbers, special characters and uppercase letters

6.2.1.4.49 ETcVnOcrOptions

Provides different options to indicate whether certain actions must be performed to achieve the intended result.

Syntax

Definition:

```
enum ETcVnOcrOptions : ULONG
{
    OO_NONE       = 0,
    OO_WITHBLANKS = 1
};
```

Values

Name	Description
OO_NONE	No options
OO_WITHBLANKS	Indicates that if blanks were found, they should be included in the result. By default, all blanks are omitted from the result.

6.2.1.4.50 ETcVnOrientationMethod

Offers methods to calculate the orientation of a set of points.

Syntax

Definition:

```
enum ETcVnOrientationMethod : LONG
{
    OM_PCA           = 0,
    OM_FITELLIPSE    = 1,
    OM_MOMENTS       = 2,
    OM_ENCLOSINGRECT = 3
};
```

Values

Name	Description
OM_PCA	Apply the PCA on the points and calculate the mean point as the center point and the rotation angle of the main principal axis
OM_FITELLIPSE	Fit an ellipse around the points and calculate the center point and the rotation angle of the main axis
OM_MOMENTS	Calculate the moments of the points and estimate the center and the rotation angle
OM_ENCLOSINGRECT	Calculate a minimum area rectangle enclosing the points and calculate the center point and the rotation angle of the rectangle

6.2.1.4.51 ETcVnPaddingMode

Offers padding modes.

Syntax

Definition:

```
enum ETcVnPaddingMode : LONG
{
    PM_NONE        = 0,
    PM_CROP_CENTER = 1,
    PM_LETTERBOX   = 2
};
```

Values

Name	Description
PM_NONE	Resize the image to the desired size without extra processing
PM_CROP_CENTER	Resize and crop the image
PM_LETTERBOX	Resize the image to the desired size while preserving the aspect ratio of the original image

6.2.1.4.52 ETcVnPixelConnectivity

Offers pixel connectivities.

Syntax

Definition:

```
enum ETcVnPixelConnectivity : LONG
{
    PC_4 = 4,
    PC_8 = 8
};
```

Values

Name	Description
PC_4	4-way
PC_8	8-way

6.2.1.4.53 ETcVnPixelEncoding

Offers pixel encodings.

Syntax

Definition:

```
enum ETcVnPixelEncoding : BYTE
{
    PE_NONE                = 0,
    PE_BAYER_GR            = 1,
    PE_BAYER_RG            = 2,
    PE_BAYER_GB            = 3,
    PE_BAYER_BG            = 4,
    PE_YUV_411_UYYVYY      = 5,
    PE_YUV_422_UYVY        = 6,
    PE_YUV_422_YUYV        = 7,
    PE_YCBCR_411_CBYYCRYY = 8,
    PE_YCBCR_422_CBYCRY    = 9,
    PE_YCBCR_422_YCBYCR    = 10
};
```

Values

Name	Description
PE_NONE	No encoding available, i.e. every pixel value is independent of other pixels.
PE_BAYER_GR	Pixels are encoded as a BayerGR pattern.
PE_BAYER_RG	Pixels are encoded as a BayerRG pattern.
PE_BAYER_GB	Pixels are encoded as a BayerGB pattern.
PE_BAYER_BG	Pixels are encoded as a BayerBG pattern.
PE_YUV_411_UYYVYY	Pixels are encoded as YUV411 (UYYVYY).
PE_YUV_422_UYVY	Pixels are encoded as YUV422 (UYVY).
PE_YUV_422_YUYV	Pixels are encoded as YUV422 (YUYV).
PE_YCBCR_411_CBYYCRYY	Pixels are encoded as YCbCr411 (CbYYCrYY).
PE_YCBCR_422_CBYCRY	Pixels are encoded as YCbCr422 (CbYCrY).
PE_YCBCR_422_YCBYCR	Pixels are encoded as YCbCr422 (YCbYCr).

6.2.1.4.54 ETcVnPixelPackMode

Offers pixel packing modes.

Syntax

Definition:

```
enum ETcVnPixelPackMode : BYTE
{
    PPM_NONE          = 0,
    PPM_MONO1P        = 1,
    PPM_MONO2P        = 2,
    PPM_MONO4P        = 3,
    PPM_MONO10PACKED  = 4,
    PPM_MONO12PACKED  = 5,
    PPM_RGB10V1PACKED = 6,
    PPM_RGB10P32      = 7,
    PPM_RGB12V1PACKED = 8,
    PPM_RGB565P       = 9,
    PPM_BGR565P       = 10,
    PPM_MONO10P       = 11,
    PPM_MONO12P       = 12,
    PPM_MONO14P       = 13
};
```

Values

Name	Description
PPM_NONE	No packing
PPM_MONO1P	Mono1p
PPM_MONO2P	Mono2p
PPM_MONO4P	Mono4p
PPM_MONO10PACKED	Mono10Packed or BayerXX10Packed
PPM_MONO12PACKED	Mono12Packed or BayerXX12Packed
PPM_RGB10V1PACKED	RGB10V1Packed
PPM_RGB10P32	RGB10V2Packed (RGB10p32)
PPM_RGB12V1PACKED	RGB12V1Packed
PPM_RGB565P	RGB565p
PPM_BGR565P	BGR565p
PPM_MONO10P	Mono10p or BayerXX10p
PPM_MONO12P	Mono12p or BayerXX12p
PPM_MONO14P	Mono14p or BayerXX14p

6.2.1.4.55 ETcVnPrototypeClusterer

Offers prototype clusterer model types.

Syntax

Definition:

```
enum ETcVnPrototypeClusterer : ULONG
{
    PC_NOVELTY_DETECTOR = 2,
    PC_CLUSTERER        = 8
};
```

Values

Name	Description
PC_NOVELTY_DETECTOR	Novelty detector
PC_CLUSTERER	Clusterer

6.2.1.4.56 ETcVnRectangleIntersection

Offers rectangle intersection types.

Syntax

Definition:

```
enum ETcVnRectangleIntersection : LONG
{
    RI_NONE    = 0,
    RI_PARTIAL = 1,
    RI_FULL    = 2
};
```

Values

Name	Description
RI_NONE	No intersection
RI_PARTIAL	Partial intersection
RI_FULL	Full intersection

6.2.1.4.57 ETcVnRotationAngle

Offers rotation angles.

Syntax

Definition:

```
enum ETcVnRotationAngle : LONG
{
    RA_90_DEG  = 0,
    RA_180_DEG = 1,
    RA_270_DEG = 2
};
```

Values

Name	Description
RA_90_DEG	90 degrees
RA_180_DEG	180 degrees
RA_270_DEG	270 degrees

6.2.1.4.58 ETcVnRTrees

Offers RTrees model types.

Syntax

Definition:

```
enum ETcVnRTrees : ULONG
{
    RT_CLASSIFIER = 1,
    RT_REGRESSOR  = 4
};
```

Values

Name	Description
RT_CLASSIFIER	Classifier
RT_REGRESSOR	Regressor

6.2.1.4.59 ETcVnSignedNormalization

Offers options for normalizing images with signed data types.

Syntax

Definition:

```
enum ETcVnSignedNormalization : SHORT
{
    SN_FIX_ZERO   = 0,
    SN_FULL_SCALE = 1
};
```

Values

Name	Description
SN_FIX_ZERO	The value 0 is fixed, i.e. the normalized image might only reach either the minimum or maximum value.
SN_FULL_SCALE	The image is normalized to the full value range, i.e. a prior value of 0 might be != 0 after normalization.

6.2.1.4.60 ETcVnSolvePnPMethod

Offers SolvePnP methods.

Syntax

Definition:

```
enum ETcVnSolvePnPMethod : LONG
{
    SPM_ITERATIVE   = 0,
    SPM_EPNP        = 1,
    SPM_P3P         = 2,
    SPM_AP3P        = 5,
    SPM_IPPE        = 6,
    SPM_IPPE_SQUARE = 7,
    SPM_SQPNP       = 8
};
```

Values

Name	Description
SPM_ITERATIVE	Levenberg-Marquardt optimization. Requires at least 4 planar or 6 non-planar reference points.
SPM_EPNP	Efficient perspective-n-point camera pose estimation. Requires at least 4 reference points.
SPM_P3P	Complete solution classification for the perspective-three-point problem. Requires exactly 4 reference points to find a unique solution.
SPM_AP3P	An efficient algebraic solution to the perspective-three-point problem. Requires exactly 4 reference points to find a unique solution.
SPM_IPPE	Infinitesimal plane-based pose estimation. Requires at least 4 coplanar reference points.
SPM_IPPE_SQUARE	IPPE specialization for squares. Requires exactly 4 reference points in the following order (hsl: half square length): (-hsl, hsl, 0), (hsl, hsl, 0), (hsl, -hsl, 0), (-hsl, -hsl, 0)
SPM_SQPNP	A consistently fast and globally optimal solution to the perspective-n-point problem. Requires at least 4 reference points.

6.2.1.4.61 ETcVnSta

Offers Simplified TopoART model types.

Syntax

Definition:

```
enum ETcVnSta : ULONG
{
    STA_CLASSIFIER       = 1,
    STA_NOVELTY_DETECTOR = 2,
    STA_REGRESSOR        = 4,
    STA_CLUSTERER        = 8
};
```

Values

Name	Description
STA_CLASSIFIER	Classifier
STA_NOVELTY_DETECTOR	Novelty detector
STA_REGRESSOR	Regressor
STA_CLUSTERER	Clusterer

6.2.1.4.62 ETcVnStructuringElementShape

Offers shapes for a structuring element.

Syntax

Definition:

```
enum ETcVnStructuringElementShape : LONG
{
    SES_RECTANGLE = 0,
    SES_CROSS     = 1,
    SES_ELLIPSE   = 2
};
```

Values

Name	Description
SES_RECTANGLE	Rectangle
SES_CROSS	Cross
SES_ELLIPSE	Ellipse

6.2.1.4.63 ETcVnSvm

Offers SVM model types.

Syntax

Definition:

```
enum ETcVnSvm : ULONG
{
    SVM_C_CLASSIFIER     = 1,
    SVM_NU_CLASSIFIER    = 65537,
    SVM_NOVELTY_DETECTOR = 2,
    SVM_EPS_REGRESSOR    = 4,
    SVM_NU_REGRESSOR     = 65540
};
```

Values

Name	Description
SVM_C_CLASSIFIER	Classifier
SVM_NU_CLASSIFIER	Classifier
SVM_NOVELTY_DETECTOR	Novelty detector
SVM_EPS_REGRESSOR	Regressor
SVM_NU_REGRESSOR	Regressor

6.2.1.4.64 ETcVnSvmKernelType

Offers SVM kernel types.

Syntax

Definition:

```
enum ETcVnSvmKernelType : LONG
{
    SKT_LINEAR  = 0,
    SKT_POLY    = 1,
    SKT_RBF     = 2,
    SKT_SIGMOID = 3,
    SKT_CHI2    = 4,
    SKT_INTER   = 5
};
```

Values

Name	Description
SKT_LINEAR	Linear kernel (fast)
SKT_POLY	Polynomial kernel
SKT_RBF	Radial basis function (RBF) kernel (good default)
SKT_SIGMOID	Sigmoid kernel
SKT_CHI2	Chi-squared kernel
SKT_INTER	Histogram intersection kernel

6.2.1.4.65 ETcVnSvmSgdClassifierMarginType

Offers different margin types for SVMSGD classifiers.

Syntax

Definition:

```
enum ETcVnSvmSgdClassifierMarginType : LONG
{
    SSCMT_SOFT_MARGIN = 0,
    SSCMT_HARD_MARGIN = 1
};
```

Values

Name	Description
SSCMT_SOFT_MARGIN	Soft margin allowing outliers. (beneficial for classes that cannot be separated linearly)
SSCMT_HARD_MARGIN	Hard margin. (best-suited for linearly separable classes)

6.2.1.4.66 ETcVnSvmSgdClassifierType

Offers different types of SVMSGD classifiers.

Syntax

Definition:

```
enum ETcVnSvmSgdClassifierType : LONG
{
    SSCT_SGD  = 0,
    SSCT_ASGD = 1
};
```

Values

Name	Description
SSCT_SGD	Stochastic Gradient Descent
SSCT_ASGD	Average Stochastic Gradient Descent

6.2.1.4.67 ETcVnTemplateMatchMethod

Offers methods for template matching.

Syntax

Definition:

```
enum ETcVnTemplateMatchMethod : LONG
{
    TMM_SQDIFF        = 0,
    TMM_SQDIFF_NORMED = 1,
    TMM_CCORR         = 2,
    TMM_CCORR_NORMED  = 3,
    TMM_CCOEFF        = 4,
    TMM_CCOEFF_NORMED = 5
};
```

Values

Name	Description
TMM_SQDIFF	Squared difference (supports template mask)
TMM_SQDIFF_NORMED	Normalized squared difference
TMM_CCORR	Cross-correlation
TMM_CCORR_NORMED	Normalized cross-correlation (supports template mask)
TMM_CCOEFF	Correlation coefficient
TMM_CCOEFF_NORMED	Normalized correlation coefficient

6.2.1.4.68 ETcVnThresholdType

Offers threshold types.

Syntax

Definition:

```
enum ETcVnThresholdType : LONG
{
    TT_BINARY              = 0,
    TT_BINARY_INV          = 1,
    TT_TRUNC               = 2,
    TT_TOZERO              = 3,
    TT_TOZERO_INV          = 4,
    TT_OTSU_BINARY         = 8,
    TT_OTSU_BINARY_INV     = 9,
    TT_OTSU_TRUNC          = 10,
    TT_OTSU_TOZERO         = 11,
    TT_OTSU_TOZERO_INV     = 12,
    TT_TRIANGLE_BINARY     = 16,
    TT_TRIANGLE_BINARY_INV = 17,
    TT_TRIANGLE_TRUNC      = 18,
    TT_TRIANGLE_TOZERO     = 19,
    TT_TRIANGLE_TOZERO_INV = 20
};
```

Values

Name	Description
TT_BINARY	Binary threshold
TT_BINARY_INV	Inverted binary threshold
TT_TRUNC	Truncated threshold (pixels > thresh are set to thresh, others keep their value)
TT_TOZERO	To zero threshold (pixels < thresh are set to zero, others keep their value)
TT_TOZERO_INV	Inverted to zero threshold (pixels > thresh are set to zero, others keep their value)
TT_OTSU_BINARY	Binary threshold with the threshold value selected according to the Otsu algorithm
TT_OTSU_BINARY_INV	Inverted binary threshold with the threshold value selected according to the Otsu algorithm
TT_OTSU_TRUNC	Truncated threshold with the threshold value selected according to the Otsu algorithm
TT_OTSU_TOZERO	To zero threshold with the threshold value selected according to the Otsu algorithm
TT_OTSU_TOZERO_INV	Inverted to zero threshold with the threshold value selected according to the Otsu algorithm
TT_TRIANGLE_BINARY	Binary threshold with the threshold value selected according to the Triangle algorithm
TT_TRIANGLE_BINARY_INV	Inverted binary threshold with the threshold value selected according to the Triangle algorithm
TT_TRIANGLE_TRUNC	Truncated threshold with the threshold value selected according to the Triangle algorithm
TT_TRIANGLE_TOZERO	To zero threshold with the threshold value selected according to the Triangle algorithm
TT_TRIANGLE_TOZERO_INV	Inverted to zero threshold with the threshold value selected according to the Triangle algorithm

6.2.1.4.69 ETcVnTimestamp

Offers image acquisition timestamps.

Syntax

Definition:

```
enum ETcVnTimestamp : ULONGLONG
{
    TS_IMAGE_COMPLETED = 1
};
```

Values

Name	Description
TS_IMAGE_COMPLETED	Timestamp when the image was completed, i.e. all packets were received by the GigEVision driver.

6.2.1.4.70 ETcVnVectorCompareMethod

Offers methods for vector comparison.

Syntax

Definition:

```
enum ETcVnVectorCompareMethod : LONG
{
    VCM_EUCLIDEAN   = 0,
    VCM_ELEMENTWISE = 1
};
```

Values

Name	Description
VCM_EUCLIDEAN	Euclidean
VCM_ELEMENTWISE	Elementwise

6.2.1.4.71 ETcVnWorldCoordinateSystem

Offers world coordinate system orientations.

Syntax

Definition:

```
enum ETcVnWorldCoordinateSystem : LONG
{
    WCS_X_RIGHT_Y_DOWN = 0,
    WCS_X_DOWN_Y_RIGHT = 1,
    WCS_X_LEFT_Y_DOWN  = 2,
    WCS_X_UP_Y_RIGHT   = 3,
    WCS_X_RIGHT_Y_UP   = 4,
    WCS_X_DOWN_Y_LEFT  = 5,
    WCS_X_LEFT_Y_UP    = 6,
    WCS_X_UP_Y_LEFT    = 7
};
```

Values

Name	Description
WCS_X_RIGHT_Y_DOWN	X axis points right, Y axis down (same orientation as image coordinate system)
WCS_X_DOWN_Y_RIGHT	X axis points down, Y axis right
WCS_X_LEFT_Y_DOWN	X axis points left, Y axis down
WCS_X_UP_Y_RIGHT	X axis points up, Y axis right
WCS_X_RIGHT_Y_UP	X axis points right, Y axis up
WCS_X_DOWN_Y_LEFT	X axis points down, Y axis left
WCS_X_LEFT_Y_UP	X axis points left, Y axis up
WCS_X_UP_Y_LEFT	X axis points up, Y axis left

6.2.1.4.72 ETcWatchdogAccumulationType

Offers watchdog accumulation types to compute the fraction processed.

Syntax

Definition:

```
enum ETcWatchdogAccumulationType : USHORT
{
    WD_ACC_TYPE_MEAN    = 1,
    WD_ACC_TYPE_PRODUCT = 2
};
```

Values

Name	Description
WD_ACC_TYPE_MEAN	Calculates the mean of the individual fractions processed over all monitored functions (recommended for independent functions).
WD_ACC_TYPE_PRODUCT	Calculates the product of the individual fractions processed over all monitored functions (recommended for dependent functions).

6.2.1.4.73 GEV_CAMERA_STATE

Describes the GigEVision camera connection state.

Syntax

Definition:

```
enum GEV_CAMERA_STATE : BYTE
{
    GEV_CAMERA_IDLE               = 0,
    GEV_CONTROL_CHANNEL_OPEN_MASK = 1,
    GEV_STREAM_CHANNELS_OPEN_MASK = 2,
    GEV_CONTROL_CHANNEL_OPEN      = 1,
    GEV_STREAM_CHANNELS_OPEN      = 3
};
```

Values

Name	Description
GEV_CAMERA_IDLE	Camera is idle (no control or stream channel open).
GEV_CONTROL_CHANNEL_OPEN_MASK	The control channel is open.
GEV_STREAM_CHANNELS_OPEN_MASK	At least 1 stream channel is open.
GEV_CONTROL_CHANNEL_OPEN	The control channel is open, but no stream channel.
GEV_STREAM_CHANNELS_OPEN	At least 1 stream channel and the control channel are open.

6.2.1.4.74 ETcVnDataLayout

Offers image data layouts.

Syntax

Definition:

```
enum ETcVnDataLayout : ULONGLONG
{
    DL_DEFAULT = 0x0,
    DL_2D_HW   = 0x12,
    DL_2D_WH   = 0x21,
    DL_3D_CHW  = 0x123,
    DL_3D_HWC  = 0x231,
    DL_4D_NCHW = 0x1234,
    DL_4D_NHWC = 0x1342
};
```

Values

Name	Description
DL_DEFAULT	Default
DL_2D_HW	2D data HW, multi/single channel image
DL_2D_WH	2D data WH, multi/single channel image
DL_3D_CHW	3D data CHW, single channel image
DL_3D_HWC	3D data HWC, single channel image
DL_4D_NCHW	4D data NCHW, single channel image
DL_4D_NHWC	4D data NHWC, single channel image

6.2.1.4.75 ETcVnReduceType

Offers reduction types.

Syntax

Definition:

```
enum ETcVnReduceType : LONG
{
    RT_REDUCE_SUM = 0,
    RT_REDUCE_AVG = 1,
    RT_REDUCE_MAX = 2,
    RT_REDUCE_MIN = 3
};
```

Values

Name	Description
RT_REDUCE_SUM	The sum (column/row-wise) of all rows/columns
RT_REDUCE_AVG	The mean (column/row-wise) of all rows/columns
RT_REDUCE_MAX	The maximum (column/row-wise) of all rows/columns
RT_REDUCE_MIN	The minimum (column/row-wise) of all rows/columns

6.2.1.4.76 ETcVnCircularityCalculationMethod

Offers methods to calculate contour circularity

Syntax

Definition:

```
enum ETcVnCircularityCalculationMethod : LONG
{
    CCM_DEFAULT                = 0,
    CCM_CONVEXHULL_PERIMETER   = 1,
    CCM_MAX_RADIUS             = 2,
    CCM_ENCLOSINGCIRCLE_RADIUS = 3
};
```

Values

Name	Description
CCM_DEFAULT	4pi * area / perimeter^2
CCM_CONVEXHULL_PERIMETER	4pi * area / (convex hull perimeter)^2
CCM_MAX_RADIUS	area / (pi * r^2), r = max distance to the contour's center of mass
CCM_ENCLOSINGCIRCLE_RADIUS	area / (pi * r^2), r = radius of the minimum enclosing circle

6.2.1.4.77 ETcVnDataMatrixOptions

Offers Data Matrix code detection options.

Syntax

Definition:

```
enum ETcVnDataMatrixOptions : ULONGLONG
{
    DMO_DEFAULT          = 0,
    DMO_BINARY_INPUT     = 1,
    DMO_GLOBAL_THRESHOLD = 2,
    DMO_BINARY_GRID      = 4,
    DMO_STATIC_GRID      = 8,
    DMO_REGULAR_COLOR    = 16,
    DMO_INVERTED_COLOR   = 32
};
```

Values

Name	Description
DMO_DEFAULT	Expects a graylevel image as input. Applies an adaptive local threshold and creates a dynamic grid based on the clock pattern.
DMO_BINARY_INPUT	Expects a binary image as input (all pixels must contain 0 or 255, disables internal preprocessing)
DMO_GLOBAL_THRESHOLD	Applies a global instead of an adaptive local threshold (requires uniform illumination)
DMO_BINARY_GRID	Create the dynamic grid based on the binary instead of the graylevel image
DMO_STATIC_GRID	Use an equally spaced static grid instead of a dynamic one
DMO_REGULAR_COLOR	Only search for regular color (dark code on light background)
DMO_INVERTED_COLOR	Only search for inverted color (light code on dark background)

6.2.1.4.78 ETcVnLayerNameType

Offer different types of neural network model layer names

Syntax

Definition:

```
enum ETcVnLayerNameType : LONG
{
    LNT_ALL     = 0,
    LNT_OUTPUTS = 1,
    LNT_INPUTS  = 2
};
```

Values

Name	Description
LNT_ALL	Return the name of all layers
LNT_OUTPUTS	Return the names of unconnected output layers
LNT_INPUTS	Return the names of model inputs

6.2.1.5 Structs

6.2.1.5.1 GVSP_IMAGE_INFO

Shows GVSP (GigE Vision Streaming Protocol) meta information.

Syntax

Definition:

```
struct GVSP_IMAGE_INFO
{
    ULONG                     CameraIpAddress;
    ULONG                     LocalIpAddress;
    USHORT                    CameraUdpPort;
    USHORT                    LocalUdpPort;
    USHORT                    GvspChannelId;
    USHORT                    GevStatus;
    ULONGLONG                 BlockId;
    GVSP_LEADER_PAYLOAD_IMAGE LeaderInfo;
};
```

Parameters

Name	Type	Description
CameraIpAddress	ULONG	Camera IP address
LocalIpAddress	ULONG	Local IP address
CameraUdpPort	USHORT	Camera UDP port
LocalUdpPort	USHORT	Local UDP port
GvspChannelId	USHORT	ID of GVSP channel
GevStatus	USHORT	GigE Vision status code
BlockId	ULONGLONG	Block Id (incremented for each acquired image, but reset to 0 on overflow)
LeaderInfo	GVSP_LEADER_PAYLOAD_IMAGE [▶ 1763]	Contains information about timestamp, pixel format and size.

6.2.1.5.2 GVSP_LEADER_PAYLOAD_IMAGE

Shows information over the GVSP (GigE Vision Streaming Protocol) leader payload image.

Syntax

Definition:

```
struct GVSP_LEADER_PAYLOAD_IMAGE
{
    ULONGLONG         Timestamp;
    GVSP_PIXEL_FORMAT PixelFormat;
    ULONG             SizeX;
    ULONG             SizeY;
    ULONG             OffsetX;
    ULONG             OffsetY;
    USHORT            PaddingX;
    USHORT            PaddingY;
};
```

Parameters

Name	Type	Description
Timestamp	ULONGLONG	Image timestamp
PixelFormat	GVSP_PIXEL_FORMAT [▶ 1764]	Image pixel format
SizeX	ULONG	Image size in x direction
SizeY	ULONG	Image size in y direction
OffsetX	ULONG	Image x-offset from (0,0) origin
OffsetY	ULONG	Image y-offset from (0,0) origin
PaddingX	USHORT	Image padding in x direction
PaddingY	USHORT	Image padding in y direction

6.2.1.5.3 GVSP_PIXEL_FORMAT

Shows the GVSP (GigE Vision Streaming Protocol) pixel format.

Syntax

Definition:

```
struct GVSP_PIXEL_FORMAT
{
    BYTE   Color;
    BYTE   EffectivePixelSize;
    USHORT Id;
};
```

Parameters

Name	Type	Description
Color	BYTE	Indicates if the pixel format is mono or color.
EffectivePixelSize	BYTE	Effective pixel size in bit
Id	USHORT	GVSP pixel format ID

6.2.1.5.4 TcVnCameraCalibrationOptions

Offers camera calibration options.

Syntax

Definition:

```
struct TcVnCameraCalibrationOptions
{
    UCHAR bUseIntrinsicGuess : 1;
    UCHAR bFixAspectRatio : 1;
    UCHAR bFixPrincipalPoint : 1;
    UCHAR bZeroTangentDist : 1;
    UCHAR bFixFocalLength : 1;
    UCHAR bFixK1 : 1;
    UCHAR bFixK2 : 1;
    UCHAR bFixK3 : 1;
    UCHAR bFixK4 : 1;
    UCHAR bFixK5 : 1;
    UCHAR bFixK6 : 1;
    UCHAR bRationalModel : 1;
};
```

Parameters

Name	Type	Description
bUseIntrinsicGuess	UCHAR : 1	The camera matrix and distortion coefficients already contain a valid initial guess, which is optimized further.
bFixAspectRatio	UCHAR : 1	The ratio fx/fy stays the same as in the input camera matrix.
bFixPrincipalPoint	UCHAR : 1	The principal point is fixed to the image center (or provided cx,cy if bUseIntrinsicGuess).
bZeroTangentDist	UCHAR : 1	The tangential distortion coefficients (p1, p2) are forced to 0.
bFixFocalLength	UCHAR : 1	The parameters fx and fy stay the same as in the input camera matrix.
bFixK1	UCHAR : 1	The radial distortion coefficient k1 is fixed to 0 (or provided input if bUseIntrinsicGuess)
bFixK2	UCHAR : 1	The radial distortion coefficient k2 is fixed to 0 (or provided input if bUseIntrinsicGuess)
bFixK3	UCHAR : 1	The radial distortion coefficient k3 is fixed to 0 (or provided input if bUseIntrinsicGuess)
bFixK4	UCHAR : 1	The radial distortion coefficient k4 is fixed to 0 (or provided input if bUseIntrinsicGuess)
bFixK5	UCHAR : 1	The radial distortion coefficient k5 is fixed to 0 (or provided input if bUseIntrinsicGuess)
bFixK6	UCHAR : 1	The radial distortion coefficient k6 is fixed to 0 (or provided input if bUseIntrinsicGuess)
bRationalModel	UCHAR : 1	The radial distortion coefficients k4, k5, k6 are enabled.

6.2.1.5.5 TcVnCircularArc

Describes a circular arc.

Syntax

Definition:

```
struct TcVnCircularArc
{
    TcVnPoint2_REAL aCenter;
    float           fRadius;
    float           fStartAngle;
    float           fEndAngle;
};
```

Parameters

Name	Type	Description
aCenter	TcVnPoint2_REAL [▶ 1711]	Center of the circular arc
fRadius	float	Radius of the circular arc
fStartAngle	float	Start angle of the circular arc
fEndAngle	float	End angle of the circular arc

6.2.1.5.6 TcVnCodeGrades1D

Describes code quality grades for 1D barcodes according to ISO / IEC 15416:2016. The grades range from 0 (very bad) to 4 (very good) and are averaged over several scan lines.

Syntax

Definition:

```
struct TcVnCodeGrades1D
{
    float fDecode;
    float fSymbolContrast;
    float fMinReflectance;
    float fMinEdgeContrast;
    float fModulation;
    float fDefects;
    float fDecodability;
    float fOverall;
};
```

Parameters

Name	Type	Description
fDecode	float	Decode grade
fSymbolContrast	float	Symbol contrast grade
fMinReflectance	float	Minimum reflectance grade
fMinEdgeContrast	float	Minimum edge contrast grade
fModulation	float	Modulation grade
fDefects	float	Defects grade
fDecodability	float	Decodability grade
fOverall	float	Overall code grade, i.e. the minimum achieved individual grade averaged over all scan lines. If different scan lines lead to different decoded data, the overall grade is 0.

6.2.1.5.7 TcVnCodeGradesDM

Describes code quality grades for Data Matrix codes according to ISO / IEC 15415:2011. The grades range from 0 (very bad) to 4 (very good).

Syntax

Definition:

```
struct TcVnCodeGradesDM
{
    unsigned char nDecode;
    unsigned char nSymbolContrast;
    unsigned char nModulation;
    unsigned char nReflectanceMargin;
    unsigned char nFixedPatternDamage;
    unsigned char nAxialNonuniformity;
```

```
    unsigned char nGridNonuniformity;
    unsigned char nUnusedErrorCorrection;
    unsigned char nOverall;
};
```

Parameters

Name	Type	Description
nDecode	unsigned char	Decode grade
nSymbolContrast	unsigned char	Symbol contrast grade
nModulation	unsigned char	Modulation grade
nReflectanceMargin	unsigned char	Reflectance margin grade
nFixedPatternDamage	unsigned char	Fixed pattern damage grade
nAxialNonuniformity	unsigned char	Axial nonuniformity grade
nGridNonuniformity	unsigned char	Grid nonuniformity grade
nUnusedErrorCorrection	unsigned char	Unused error correction grade
nOverall	unsigned char	Overall code grade, i.e. the minimum achieved individual grade

6.2.1.5.8 TcVnCodeGradesQR

Describes code quality grades for QR codes according to ISO / IEC 15415:2011. The grades range from 0 (very bad) to 4 (very good).

Syntax

Definition:

```
struct TcVnCodeGradesQR
{
    unsigned char nDecode;
    unsigned char nSymbolContrast;
    unsigned char nModulation;
    unsigned char nReflectanceMargin;
    unsigned char nFixedPatternDamage;
    unsigned char nAxialNonuniformity;
    unsigned char nGridNonuniformity;
    unsigned char nUnusedErrorCorrection;
    unsigned char nFormatInfo;
    unsigned char nVersionInfo;
    unsigned char nOverall;
};
```

Parameters

Name	Type	Description
nDecode	unsigned char	Decode grade
nSymbolContrast	unsigned char	Symbol contrast grade
nModulation	unsigned char	Modulation grade
nReflectanceMargin	unsigned char	Reflectance margin grade
nFixedPatternDamage	unsigned char	Fixed pattern damage grade
nAxialNonuniformity	unsigned char	Axial nonuniformity grade
nGridNonuniformity	unsigned char	Grid nonuniformity grade
nUnusedErrorCorrection	unsigned char	Unused error correction grade
nFormatInfo	unsigned char	Format information grade
nVersionInfo	unsigned char	Version information grade
nOverall	unsigned char	Overall code grade, i.e. the minimum achieved individual grade

6.2.1.5.9 TcVnDMatch

Describes a descriptor match

Syntax

Definition:

```
struct TcVnDMatch
{
    LONG  nQueryIdx;
    LONG  nTrainIdx;
    LONG  nImageIdx;
    float fDistance;
};
```

Parameters

Name	Type	Description
nQueryIdx	LONG	Query descriptor index
nTrainIdx	LONG	Train descriptor index
nImageIdx	LONG	Train image index
fDistance	float	Distance between the descriptors (smaller distance means better match)

6.2.1.5.10 TcVnImageInfo

Shows image information.

Syntax

Definition:

```
struct TcVnImageInfo
{
    ULONGLONG       nImageSize;
    ULONG           nWidth;
    ULONG           nHeight;
    USHORT          nXPadding;
    USHORT          nYPadding;
    TcVnPixelFormat stPixelFormat;
};
```

Parameters

Name	Type	Description
nImageSize	ULONGLONG	Image size (number of pixels)
nWidth	ULONG	Image width
nHeight	ULONG	Image height
nXPadding	USHORT	Image x-padding
nYPadding	USHORT	Image y-padding
stPixelFormat	TcVnPixelFormat [▶ 1780]	Pixel format

6.2.1.5.11 TcVnKeyPoint

Describes a key point.

Syntax

Definition:

```
struct TcVnKeyPoint
{
    TcVnPoint2_REAL aPoint;
    float           fDiameter;
    float           fAngle;
    float           fResponse;
    LONG            nOctave;
    LONG            nClassId;
};
```

Parameters

Name	Type	Description
aPoint	TcVnPoint2_REAL [▶ 1711]	Position
fDiameter	float	Diameter
fAngle	float	Angle
fResponse	float	Response
nOctave	LONG	Octave
nClassId	LONG	Class ID

6.2.1.5.12 TcVnMatrix

Offers a user-defined matrix with variable rows, columns and element-type.

Syntax

Definition:

```
struct TcVnMatrix
{
    ULONG            nRows;
    ULONG            nCols;
    ETcVnElementType eType;
    PVOID            pData;
};
```

Parameters

Name	Type	Description
nRows	ULONG	Number of rows
nCols	ULONG	Number of columns
eType	ETcVnElementType [▶ 1738]	Element type
pData	PVOID	Pointer to the data

6.2.1.5.13 TcVnMoments

Offers image or contour moments.

Syntax

Definition:

```
struct TcVnMoments
{
    double fM00;
    double fM10;
    double fM01;
    double fM20;
    double fM11;
    double fM02;
    double fM30;
    double fM21;
    double fM12;
    double fM03;
    double fMu20;
    double fMu11;
```

```
    double fMu02;
    double fMu30;
    double fMu21;
    double fMu12;
    double fMu03;
    double fNu20;
    double fNu11;
    double fNu02;
    double fNu30;
    double fNu21;
    double fNu12;
    double fNu03;
};
```

Parameters

Name	Type	Description
fM00	double	Spatial moment 00
fM10	double	Spatial moment 10
fM01	double	Spatial moment 01
fM20	double	Spatial moment 20
fM11	double	Spatial moment 11
fM02	double	Spatial moment 02
fM30	double	Spatial moment 30
fM21	double	Spatial moment 21
fM12	double	Spatial moment 12
fM03	double	Spatial moment 03
fMu20	double	Central moment 20
fMu11	double	Central moment 11
fMu02	double	Central moment 02
fMu30	double	Central moment 30
fMu21	double	Central moment 21
fMu12	double	Central moment 12
fMu03	double	Central moment 03
fNu20	double	Normalized central moment 20
fNu11	double	Normalized central moment 11
fNu02	double	Normalized central moment 02
fNu30	double	Normalized central moment 30
fNu21	double	Normalized central moment 21
fNu12	double	Normalized central moment 12
fNu03	double	Normalized central moment 03

6.2.1.5.14 TcVnParamsAGAST

Offers parameters for AGAST method.

Syntax

Definition:

```
struct TcVnParamsAGAST
{
    LONG                            nThreshold;
    bool                            bNonMaxSuppression;
    ETcVnKeypointDetectionTypeAGAST eType;
};
```

Parameters

Name	Type	Default	Description
nThreshold	LONG	10	Threshold for the intensity difference between the center pixel and its surrounding circle
bNonMaxSuppression	bool	true	If true, non-maximum suppression is applied.
eType	ETcVnKeypointDetectionTypeAGAST [▶ 1744]	KDT_AGAST_9_16	Neighborhood type

6.2.1.5.15 TcVnParamsAKAZE

Offers parameters for AKAZE method.

Syntax

Definition:

```
struct TcVnParamsAKAZE
{
    ETcVnFeatureDescriptorTypeAKAZE eDescrType;
    ULONG                           nDescrSize;
    ULONG                           nDescrChannels;
    float                           fThreshold;
    ULONG                           nOctaves;
    ULONG                           nOctaveLayers;
    ETcVnDiffusivityTypeKAZE        eDiffusivity;
};
```

Parameters

Name	Type	Default	Description
eDescrType	ETcVnFeatureDescriptorTypeAKAZE [▶ 1740]	FDT_AKAZE_MLDB	Type of the descriptor
nDescrSize	ULONG	0	Size of the descriptor in bits (only for MLDB; 0 = full size)
nDescrChannels	ULONG	3	Number of descriptor channels (currently only 3 supported for MLDB)
fThreshold	float	0.001f	Detector response threshold
nOctaves	ULONG	2	Maximum octave evolution
nOctaveLayers	ULONG	1	Number of sublevels per scale level
eDiffusivity	ETcVnDiffusivityTypeKAZE [▶ 1735]	DT1_KAZE_PM_G2	Diffusivity type

6.2.1.5.16 TcVnParamsBlobDetection

Offers parameters for blob detection

Syntax

Definition:

```
struct TcVnParamsBlobDetection
{
    bool                bFilterByArea;
    bool                bFilterByCircularity;
    bool                bFilterByConvexity;
    bool                bFilterByEccentricity;
    bool                bFilterByInertiaRatio;
    float               fMinArea;
    float               fMaxArea;
    float               fMinCircularity;
    float               fMaxCircularity;
    float               fMinConvexity;
    float               fMaxConvexity;
    float               fMinEccentricity;
    float               fMaxEccentricity;
    float               fMinInertiaRatio;
    float               fMaxInertiaRatio;
    ETcVnThresholdType  eThresholdType;
    float               fMinThreshold;
    float               fMaxThreshold;
    float               fThresholdStep;
    float               fMinBlobDistance;
    ULONG               nMinRepeatability;
    ETcVnBlobCombination eBlobCombination;
    bool                bExcludeBorderBlobs;
};
```

Parameters

Name	Type	Default	Description
bFilterByArea	bool	true	Enable filtering by area (fMinArea <= area <= fMaxArea); strongly recommended to activate for filtering noise with fMinArea.
bFilterByCircularity	bool	false	Enable filtering by circularity (fMinCircularity <= circularity(4pi * area / perimeter^2) <= fMaxCircularity).
bFilterByConvexity	bool	false	Enable filtering by convexity (fMinConvexity <= convexity(area / hullArea) <= fMaxConvexity).
bFilterByEccentricity	bool	false	Enable filtering by eccentricity (fMinEccentricity <= eccentricity <= fMaxEccentricity).
bFilterByInertiaRatio	bool	false	Enable filtering by inertia ratio (fMinInertiaRatio <= inertia ratio <= fMaxInertiaRatio).
fMinArea	float	10	Min estimated blob area in pixel
fMaxArea	float	100000000	Max estimated blob area in pixel
fMinCircularity	float	0	Min circularity (1.0: ideal circle, < 1: less circular, 0: not circular at all)
fMaxCircularity	float	1	Max circularity (1.0: ideal circle, < 1: less circular, 0: not circular at all)
fMinConvexity	float	0	Min convexity (1.0: blob fully convex, < 1: less convex)
fMaxConvexity	float	1	Max convexity (1.0: blob fully convex, < 1: less convex)
fMinEccentricity	float	0	Min eccentricity (0.0: circular, 1.0: linear)
fMaxEccentricity	float	1	Max eccentricity (0.0: circular, 1.0: linear)
fMinInertiaRatio	float	0	Min inertia ratio (1.0: equal width and height, 0.0: linear)
fMaxInertiaRatio	float	1	Max inertia ratio (1.0: equal width and height, 0.0: linear)
eThresholdType	ETcVnThresholdType [▶ 1757]	TT_BINARY	Threshold type for internally applied threshold method (OTSU_XXX only supported for USINT images).

Name	Type	Default	Description
fMinThreshold	float	30	Threshold to start with (if fThresholdStep > 0, otherwise this is the only threshold used).
fMaxThreshold	float	225	Threshold to end with (if fThresholdStep > 0, otherwise this value is not used).
fThresholdStep	float	0	Sets to 0 if only 1 threshold should be used (much faster than multiple thresholds and combining the results).
fMinBlobDistance	float	5	Minimum distance between the center points of two different blobs (only used if fThresholdStep > 0; if distance < fMinBlobDistance, the blobs are treated as the same).
nMinRepeatability	ULONG	2	Minimum number of threshold steps, for which the same contour has to be detected (only used if fThresholdStep > 0; same means center point distance < fMinBlobDistance).
eBlobCombination	ETcVnBlobCombination [▶ 1715]	BC_MEDIAN_THRESHOLD	Selects, which of the multi-threshold blob contours should be returned.
bExcludeBorderBlobs	bool	false	If true, blobs touching the image border are excluded from the results.

6.2.1.5.17 TcVnParamsBRISK

Offers parameters for BRISK method.

Syntax

Definition:

```
struct TcVnParamsBRISK
{
    LONG  nThreshold;
    ULONG nOctaves;
    float fPatternScale;
};
```

Parameters

Name	Type	Default	Description
nThreshold	LONG	30	Detection threshold
nOctaves	ULONG	3	Detection octaves (0 for single scale)
fPatternScale	float	1	Scale factor for the neighborhood pattern

6.2.1.5.18 TcVnParamsFAST

Offers parameters for FAST method.

Syntax

Definition:

```
struct TcVnParamsFAST
{
    LONG                           nThreshold;
    bool                           bNonMaxSuppression;
    ETcVnKeypointDetectionTypeFAST eType;
};
```

Parameters

Name	Type	Default	Description
nThreshold	LONG	10	Threshold for the intensity difference between the center pixel and its surrounding circle
bNonMaxSuppression	bool	true	If true, non-maximum suppression is applied.
eType	ETcVnKeypointDetectionTypeFAST [▶ 1745]	KDT_FAST_9_16	Neighborhood type

6.2.1.5.19 TcVnParamsGFTT

Offers parameters for GFTT method.

Syntax

Definition:

```
struct TcVnParamsGFTT
{
    ULONG  nMaxCorners;
    double fQualityLevel;
    double fMinDistance;
    ULONG  nBlockSize;
    bool   bUseHarrisDetector;
    double fHarrisK;
};
```

Parameters

Name	Type	Default	Description
nMaxCorners	ULONG	1000	Maximum number of corners to return (strongest ones)
fQualityLevel	double	0.01	Minimum accepted corner quality, relative to the strongest one
fMinDistance	double	1	Minimum euclidean distance between returned keypoints
nBlockSize	ULONG	3	Neighborhood block size
bUseHarrisDetector	bool	false	If true, a Harris detector is used instead of the default method.
fHarrisK	double	0.04	Free parameter of the Harris detector

6.2.1.5.20 TcVnParamsKAZE

Offers parameters for KAZE method

Syntax

Definition:

```
struct TcVnParamsKAZE
{
    bool                    bExtended;
    bool                    bUpright;
    float                   fThreshold;
    ULONG                   nOctaves;
    ULONG                   nOctaveLayers;
    ETcVnDiffusivityTypeKAZE eDiffusivity;
};
```

Parameters

Name	Type	Default	Description
bExtended	bool	false	If true, the descriptor size is extended from the default 64 byte to 128 byte.
bUpright	bool	false	If true, upright (rotation dependent) descriptors are computed.
fThreshold	float	0.001f	Detector response threshold
nOctaves	ULONG	4	Maximum octave evolution of the image
nOctaveLayers	ULONG	2	Number of sublevels per scale level
eDiffusivity	ETcVnDiffusivityTypeKAZE [▶ 1735]	DT1_KAZE_PM_G2	Diffusivity type

6.2.1.5.21 TcVnParamsMSER

Offers parameters for MSER method

Syntax

Definition:

```
struct TcVnParamsMSER
{
    LONG   nDelta;
    ULONG  nMinArea;
    ULONG  nMaxArea;
    double fMaxVariation;
    double fMinDiversity;
    ULONG  nMaxEvolution;
    double fAreaThreshold;
    double fMinMargin;
    ULONG  nEdgeBlurSize;
};
```

Parameters

Name	Type	Default	Description
nDelta	LONG	5	Delta for size comparison
nMinArea	ULONG	60	Minimum blob size
nMaxArea	ULONG	14400	Maximum blob size
fMaxVariation	double	0.25	Maximum blob size variation
fMinDiversity	double	0.2	Minimum MSER diversity
nMaxEvolution	ULONG	200	Maximum evolution steps (only used for color images)
fAreaThreshold	double	1.01	Area threshold to cause re-initialization (only used for color images)
fMinMargin	double	0.003	Minimum margin (only used for color images)
nEdgeBlurSize	ULONG	5	Aperture size for edge bluring (only used for color images)

6.2.1.5.22 TcVnParamsORB

Offers parameters for ORB method.

Syntax

Definition:

```
struct TcVnParamsORB
{
    ULONG                     nMaxPoints;
    float                     fPyramidScale;
    ULONG                     nPyramidLevels;
    ULONG                     nEdgeThreshold;
    ULONG                     nFirstLevel;
    ULONG                     nBriefPoints;
    ETcVnFeatureScoreTypeORB  eScoreType;
    ULONG                     nPatchSize;
    LONG                      nFastThreshold;
};
```

Parameters

Name	Type	Default	Description
nMaxPoints	ULONG	500	Maximum number of returned keypoints
fPyramidScale	float	1.2f	Pyramid decimation ratio (must be greater than 1, should be smaller than 2)
nPyramidLevels	ULONG	8	Number of pyramid levels
nEdgeThreshold	ULONG	31	Size of the border, where no features are detected (should match nPatchSize)
nFirstLevel	ULONG	0	First pyramid level (currently, only 0 is supported)
nBriefPoints	ULONG	2	Number of points to produce each BRIEF descriptor element (2, 3, 4)
eScoreType	ETcVnFeatureScoreTypeORB [▶ 1741]	FST_ORB_HARRIS	Score type (HARRIS is more stable but slightly slower than FAST)
nPatchSize	ULONG	31	Patch size of the BRIEF descriptor
nFastThreshold	LONG	20	Threshold for the FAST keypoint detection

6.2.1.5.23 TcVnParamsSB

Offers parameters for SB method (a simple blob detector with multiple thresholds).

Syntax

Definition:

```
struct TcVnParamsSB
{
    bool          bFilterByArea;
    bool          bFilterByCircularity;
    bool          bFilterByColor;
    bool          bFilterByConvexity;
    bool          bFilterByInertia;
    float         fMinArea;
    float         fMaxArea;
    float         fMinCircularity;
    float         fMaxCircularity;
    unsigned char nBlobColor;
    float         fMinConvexity;
    float         fMaxConvexity;
    float         fMinInertiaRatio;
    float         fMaxInertiaRatio;
    float         fMinBlobDist;
    ULONG         nMinRepeatability;
    float         fMinThreshold;
    float         fMaxThreshold;
    float         fThresholdStep;
};
```

Parameters

Name	Type	Default	Description
bFilterByArea	bool	true	Enable filtering by area (fMinArea <= area < fMaxArea).
bFilterByCircularity	bool	false	Enable filtering by circularity (fMinCircularity <= circularity(4pi * area / perimeter^2) < fMaxCircularity).
bFilterByColor	bool	false	Enable filtering by color (thresholdedColor(0 or 255) = nBlobColor).
bFilterByConvexity	bool	false	Enable filtering by convexity (fMinConvexity <= convexity(area / hullArea) < fMaxConvexity).
bFilterByInertia	bool	false	Enable filtering by inertia ratio (fMinInertiaRatio <= inertia ratio < fMaxInertiaRatio).
fMinArea	float	25	Min estimated blob area in pixel
fMaxArea	float	15000	Max estimated blob area in pixel
fMinCircularity	float	0	Min circularity (1.0: ideal circle, < 1: less circular, 0: not circular at all)
fMaxCircularity	float	1	Max circularity (1.0: ideal circle, < 1: less circular, 0: not circular at all)
nBlobColor	unsigned char	255	0 or 255
fMinConvexity	float	0	Min convexity (1.0: blob fully convex, < 1: less convex)
fMaxConvexity	float	1	Max convexity (1.0: blob fully convex, < 1: less convex)
fMinInertiaRatio	float	0	Min inertia ratio (0.0 .. 1.0)
fMaxInertiaRatio	float	1	Max inertia ratio (0.0 .. 1.0)
fMinBlobDist	float	5	Min distance between different blobs
nMinRepeatability	ULONG	2	Min number of same detected blobs (dist < fMinBlobDist) in different thresholds
fMinThreshold	float	30	Min threshold (start)
fMaxThreshold	float	225	Max threshold (stop)
fThresholdStep	float	10	Threshold step between min and max threshold

6.2.1.5.24 TcVnPixelFormat

Contains detailed information about the pixel format.

Syntax

Definition:

```
struct TcVnPixelFormat
{
    UCHAR              bSupported : 1;
    UCHAR              bSigned : 1;
    UCHAR              bPlanar : 1;
    UCHAR              bFloat : 1;
    BYTE               nChannels;
    ETcVnPixelEncoding ePixelEncoding;
    ETcVnPixelPackMode ePixelPackMode;
    USHORT             nElementSize;
    USHORT             nTotalSize;
};
```

Parameters

Name	Type	Description
bSupported	UCHAR : 1	If false, the pixel format is not supported.
bSigned	UCHAR : 1	If true, pixel intensities are signed values.
bPlanar	UCHAR : 1	If true, the image channels are stored planar instead of interleaved (e.g. RRRRRGGGGGBBBBB instead of RGBRGBRGBRGBRGB).
bFloat	UCHAR : 1	If true, the pixel format is floating point.
nChannels	BYTE	Number of channels
ePixelEncoding	ETcVnPixelEncoding [▶ 1750]	Pixel encoding
ePixelPackMode	ETcVnPixelPackMode [▶ 1751]	Pixel pack mode
nElementSize	USHORT	Size (bit) of a single pixel channel
nTotalSize	USHORT	Size (bit) of all pixel channels

6.2.1.5.25 TcVnRectangle_DINT

Contains origin and size of a rectangle.

Syntax

Definition:

```
struct TcVnRectangle_DINT
{
    LONG nX;
    LONG nY;
    LONG nWidth;
    LONG nHeight;
};
```

Parameters

Name	Type	Description
nX	LONG	X coordinate of the top-left corner
nY	LONG	Y coordinate of the top-left corner
nWidth	LONG	Width
nHeight	LONG	Height

6.2.1.5.26 TcVnRectangle_UDINT

Contains origin and size of a rectangle.

Syntax

Definition:

```
struct TcVnRectangle_UDINT
{
    ULONG nX;
    ULONG nY;
    ULONG nWidth;
    ULONG nHeight;
};
```

Parameters

Name	Type	Description
nX	ULONG	X coordinate of the top-left corner
nY	ULONG	Y coordinate of the top-left corner
nWidth	ULONG	Width
nHeight	ULONG	Height

6.2.1.5.27 TcVnRotatedRectangle

Contains center, size and angle of a rotated rectangle.

Syntax

Definition:

```
struct TcVnRotatedRectangle
{
    TcVnPoint2_REAL aCenter;
    TcVnSize2_REAL  stSize;
    float           fAngle;
};
```

Parameters

Name	Type	Description
aCenter	TcVnPoint2_REAL [▶ 1711]	Center point
stSize	TcVnSize2_REAL [▶ 1782]	Size composed of fWidth and fHeight
fAngle	float	Angle in degree

6.2.1.5.28 TcVnSize2_REAL

Contains width and height.

Syntax

Definition:

```
struct TcVnSize2_REAL
{
    float fWidth;
    float fHeight;
};
```

Parameters

Name	Type	Description
fWidth	float	Width
fHeight	float	Height

6.2.2 Interfaces

Interfaces erfordern keine Lizenz.

Gruppierung von Interfaces

Alle Interfaces der TwinCAT Vision Bibliothek sind thematisch in die folgenden Gruppen sortiert:

- Bilder [▶ 1895] ()
- Container [▶ 1783] ()
- Machine Learning [▶ 1931] ()
- Sonstiges [▶ 1934] ()

6.2.2.1 Containers

Diese Gruppe enthält Interfaces zur Handhabung von Containern [▶ 163].

Generelle Informationen

Ein Container ist grundsätzlich ein Vektor von mehreren Elementen. Dabei kann die Anzahl der Elemente dynamisch verändert werden. Dafür wird automatisch entsprechender Speicher im Router-Speicher [▶ 56] allokiert und freigegeben.

Container-Typen

Alle Elemente eines Containers haben denselben Typ. Auf Basis des Typs der enthaltenen Elemente werden Container durch eine eindeutige GUID charakterisiert. Zur einfacheren Handhabung sind diese GUIDs durch Konstanten [▶ 171] definiert. Alle Typ-Bezeichnungen für Container beginnen mit „ContainerType_Vector_“ (mit der Ausnahme „ContainerType_String_SINT“). Für alle API-Elemente, die Gebrauch von Containern machen, sind Typ-Anforderungen ggfs. dokumentiert.

Der Typ eines Containers kann folgendermaßen in Erfahrung gebracht werden:

```
ipContainer.GetElementTypeGuid(nTypeGuid);
```

6.2.2.1.1 Function Blocks

Diese Gruppe enthält Interfaces zur Implementierung bestimmter Funktionsblöcke.

6.2.2.1.1.1 ITcVnCustomContainerOperation_ITcVnContainer

Offers an interface for custom, elementwise container operations (for containers of containers) with up to 3 different containers.

Inheritance Hierarchy

ITcVnCustomContainerOperation_ITcVnContainer

Methods

Name	Origin	Description
Execute [▶ 1784]	ITcVnCustomContainerOperation_ITcVnContainer	Executes the custom operation on the container elements.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.1.1.1 Execute

Executes the custom operation on the container elements.

Syntax

Definition:

```
HRESULT Execute(
    ITcVnContainer* ipElement1,
    ITcVnContainer* ipElement2,
    ITcVnContainer* ipElement3
)
```

Parameters

Name	Type	Description
ipElement1	ITcVnContainer* [▶ 1890]	Current element of the 1st container.
ipElement2	ITcVnContainer* [▶ 1890]	Current element of the 2nd container.
ipElement3	ITcVnContainer* [▶ 1890]	Current element of the 3rd container.

Return value

HRESULT [▶ 150]

6.2.2.1.1.2 ITcVnCustomContainerOperation_ITcVnForwardIterator

Offers an interface for custom, elementwise container operations with up to 3 different containers.

Inheritance Hierarchy

ITcVnCustomContainerOperation_ITcVnForwardIterator

Methods

Name	Origin	Description
Execute [▶ 1785]	ITcVnCustomContainerOperation_ITcVnForwardIterator	Executes the custom operation on the container elements.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.1.2.1 Execute

Executes the custom operation on the container elements.

Syntax

Definition:

```
HRESULT Execute(
    ITcVnForwardIterator* ipElement1,
    ITcVnForwardIterator* ipElement2,
    ITcVnForwardIterator* ipElement3
)
```

Parameters

Name	Type	Description
ipElement1	ITcVnForwardIterator* [▶ 1881]	Current element of the 1st container.
ipElement2	ITcVnForwardIterator* [▶ 1881]	Current element of the 2nd container.
ipElement3	ITcVnForwardIterator* [▶ 1881]	Current element of the 3rd container.

Return value

HRESULT [▶ 150]

6.2.2.1.1.3 ITcVnCustomElementCondition_ITcVnContainer

Offers an interface for a custom condition computed for a container.

Inheritance Hierarchy

ITcVnCustomElementCondition_ITcVnContainer

Methods

Name	Origin	Description
Condition [▶ 1785]	ITcVnCustomElementCondition_ITcVnContainer	Evaluates the condition for the container.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.1.3.1 Condition

Evaluates the condition for the container.

Syntax

Definition:

```
bool Condition(
    ITcVnContainer* ipElement
)
```

Parameters

Name	Type	Description
ipElement	ITcVnContainer* [▶ 1890]	The container for which the condition is evaluated.

Return value

bool

6.2.2.1.1.4 ITcVnCustomElementCondition_ITcVnForwardIterator

Offers an interface for a custom condition computed for an element represented by an ITcVnForwardIterator.

Inheritance Hierarchy

ITcVnCustomElementCondition_ITcVnForwardIterator

Methods

Name	Origin	Description
Condition [▶ 1786]	ITcVnCustomElementCondition_ITcVnForwardIterator	Evaluates the condition for the provided element.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.1.4.1 Condition

Evaluates the condition for the provided element.

Syntax

Definition:

```
bool Condition(
    ITcVnForwardIterator* ipElement
)
```

Parameters

Name	Type	Description
ipElement	ITcVnForwardIterator* [▶ 1881]	Element for which the condition is evaluated.

Return value

bool

6.2.2.1.2 ITcVnAccess

Das Interface `ITcVnAccess` dient dem Zugriff auf Elemente eines Containers [▶ 164] und wird von einem Iterator [▶ 1880] bereitgestellt. Im Gegensatz zu ITcVnRandomAccess [▶ 1831] kann nur das Element adressiert werden, auf das der zugehörige Iterator in dem Moment zeigt.

6.2.2.1.2.1 ITcVnAccess_DINT

Offers an access interface for DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1787]	ITcVnAccess_DINT	Gets the value
Set [▶ 1787]	ITcVnAccess_DINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.1.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    LONG& nValue
)
```

Parameters

Name	Type	Description
nValue	LONG&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.1.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    LONG nValue
)
```

Parameters

Name	Type	Description
nValue	LONG	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.2 ITcVnAccess_INT

Offers an access interface for INT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_INT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1788]	ITcVnAccess_INT	Gets the value
Set [▶ 1789]	ITcVnAccess_INT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.2.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    SHORT& nValue
)
```

Parameters

Name	Type	Description
nValue	SHORT&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.2.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    SHORT nValue
)
```

Parameters

Name	Type	Description
nValue	SHORT	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.3 ITcVnAccess_ITcVnImage

Offers an access interface for images.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_ITcVnImage

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1789]	ITcVnAccess_ITcVnImage	Gets the image.
Set [▶ 1790]	ITcVnAccess_ITcVnImage	Sets the image.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.3.1 Get

Gets the image.

Syntax

Definition:

```
HRESULT Get(
    ITcVnImage*& ipImage
)
```

Parameters

Name	Type	Description
ipImage	ITcVnImage* [▶ 1925]&	Returns the image.

Return value

HRESULT [▶ 150]

6.2.2.1.2.3.2 Set

Sets the image.

Syntax

Definition:

```
HRESULT Set(
    ITcVnImage* ipImage
)
```

Parameters

Name	Type	Description
ipImage	ITcVnImage* [▶ 1925]	The image to set.

Return value

HRESULT [▶ 150]

6.2.2.1.2.4 ITcVnAccess_LREAL

Offers an access interface for LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1791]	ITcVnAccess_LREAL	Gets the value
Set [▶ 1791]	ITcVnAccess_LREAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.4.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    double& fValue
)
```

Parameters

Name	Type	Description
fValue	double&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.4.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    double fValue
)
```

Parameters

Name	Type	Description
fValue	double	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.5 ITcVnAccess_REAL

Offers an access interface for REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1792]	ITcVnAccess_REAL	Gets the value
Set [▶ 1792]	ITcVnAccess_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.5.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    float& fValue
)
```

Parameters

Name	Type	Description
fValue	float&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.5.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    float fValue
)
```

Parameters

Name	Type	Description
fValue	float	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.6 ITcVnAccess_SINT

Offers an access interface for SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1793]	ITcVnAccess_SINT	Gets the value
Set [▶ 1793]	ITcVnAccess_SINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.6.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    char& nValue
)
```

Parameters

Name	Type	Description
nValue	char&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.6.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    char nValue
)
```

Parameters

Name	Type	Description
nValue	char	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.7 ITcVnAccess_TcVnDMatch

Offers an access interface for TcVnDMatch values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnDMatch

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1794]	ITcVnAccess_TcVnDMatch	Gets the value
Set [▶ 1795]	ITcVnAccess_TcVnDMatch	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.7.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnDMatch& stDMatch
)
```

Parameters

Name	Type	Description
stDMatch	TcVnDMatch [▶ 1768]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.7.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    TcVnDMatch& stDMatch
)
```

Parameters

Name	Type	Description
stDMatch	TcVnDMatch [▶ 1768]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.8 ITcVnAccess_TcVnKeyPoint

Offers an access interface for TcVnKeyPoint values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnKeyPoint

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1795]	ITcVnAccess_TcVnKeyPoint	Gets the value.
Set [▶ 1796]	ITcVnAccess_TcVnKeyPoint	Sets the value.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.8.1 Get

Gets the value.

Syntax

Definition:

```
HRESULT Get(
    TcVnKeyPoint& stKeyPoint
)
```

Parameters

Name	Type	Description
stKeyPoint	TcVnKeyPoint [▶ 1768]&	Returns the value.

Return value

HRESULT [▶ 150]

6.2.2.1.2.8.2 Set

Sets the value.

Syntax

Definition:

```
HRESULT Set(
    TcVnKeyPoint& stKeyPoint
)
```

Parameters

Name	Type	Description
stKeyPoint	TcVnKeyPoint [▶ 1768]&	The value to set.

Return value

HRESULT [▶ 150]

6.2.2.1.2.9 ITcVnAccess_TcVnPoint2_DINT

Offers an access interface for TcVnPoint2_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnPoint2_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1797]	ITcVnAccess_TcVnPoint2_DINT	Gets the value.
Set [▶ 1797]	ITcVnAccess_TcVnPoint2_DINT	Sets the value.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.9.1 Get

Gets the value.

Syntax

Definition:

```
HRESULT Get(
    TcVnPoint2_DINT& aPoint
)
```

Parameters

Name	Type	Description
aPoint	TcVnPoint2_DINT [▶ 1711]&	Returns the value.

Return value

HRESULT [▶ 150]

6.2.2.1.2.9.2 Set

Sets the value.

Syntax

Definition:

```
HRESULT Set(
    TcVnPoint2_DINT& aPoint
)
```

Parameters

Name	Type	Description
aPoint	TcVnPoint2_DINT [▶ 1711]&	The value to set.

Return value

HRESULT [▶ 150]

6.2.2.1.2.10 ITcVnAccess_TcVnPoint2_LREAL

Offers an access interface for TcVnPoint2_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnPoint2_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1798]	ITcVnAccess_TcVnPoint2_LREAL	Gets the value
Set [▶ 1798]	ITcVnAccess_TcVnPoint2_LREAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.10.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnPoint2_LREAL& aPoint
)
```

Parameters

Name	Type	Description
aPoint	TcVnPoint2_LREAL [▶ 1711]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.10.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    TcVnPoint2_LREAL& aPoint
)
```

Parameters

Name	Type	Description
aPoint	TcVnPoint2_LREAL [▶ 1711]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.11 ITcVnAccess_TcVnPoint2_REAL

Offers an access interface for TcVnPoint2_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnPoint2_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1799]	ITcVnAccess_TcVnPoint2_REAL	Gets the value
Set [▶ 1799]	ITcVnAccess_TcVnPoint2_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.11.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnPoint2_REAL& aPoint
)
```

Parameters

Name	Type	Description
aPoint	TcVnPoint2_REAL [▶ 1711]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.11.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    TcVnPoint2_REAL& aPoint
)
```

Parameters

Name	Type	Description
aPoint	TcVnPoint2_REAL [▶ 1711]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.12 ITcVnAccess_TcVnPoint3_LREAL

Offers an access interface for TcVnPoint3_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnPoint3_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1800]	ITcVnAccess_TcVnPoint3_LREAL	Gets the value
Set [▶ 1801]	ITcVnAccess_TcVnPoint3_LREAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.12.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnPoint3_LREAL& aPoint
)
```

Parameters

Name	Type	Description
aPoint	TcVnPoint3_LREAL [▶ 1711]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.12.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    TcVnPoint3_LREAL& aPoint
)
```

Parameters

Name	Type	Description
aPoint	TcVnPoint3_LREAL [▶ 1711]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.13 ITcVnAccess_TcVnPoint3_REAL

Offers an access interface for TcVnPoint3_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnPoint3_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1801]	ITcVnAccess_TcVnPoint3_REAL	Gets the value
Set [▶ 1802]	ITcVnAccess_TcVnPoint3_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.13.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnPoint3_REAL& aPoint
)
```

Parameters

Name	Type	Description
aPoint	TcVnPoint3_REAL [▶ 1711]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.13.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    TcVnPoint3_REAL& aPoint
)
```

Parameters

Name	Type	Description
aPoint	TcVnPoint3_REAL [▶ 1711]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.14 ITcVnAccess_TcVnRectangle_DINT

Offers an access interface for TcVnRectangle_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnRectangle_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1803]	ITcVnAccess_TcVnRectangle_DINT	Gets the value
Set [▶ 1803]	ITcVnAccess_TcVnRectangle_DINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.14.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnRectangle_DINT& stRectangle
)
```

Parameters

Name	Type	Description
stRectangle	TcVnRectangle_DINT [▶ 1781]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.14.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    TcVnRectangle_DINT& stRectangle
)
```

Parameters

Name	Type	Description
stRectangle	TcVnRectangle_DINT [▶ 1781]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.15 ITcVnAccess_TcVnVector2_DINT

Offers an access interface for TcVnVector2_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector2_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1804]	ITcVnAccess_TcVnVector2_DINT	Gets the value
Set [▶ 1804]	ITcVnAccess_TcVnVector2_DINT	Sets the value.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.15.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnVector2_DINT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector2_DINT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.15.2 Set

Sets the value.

Syntax

Definition:

```
HRESULT Set(
    TcVnVector2_DINT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector2_DINT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.16 ITcVnAccess_TcVnVector2_INT

Offers an access interface for TcVnVector2_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector2_INT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1805]	ITcVnAccess_TcVnVector2_INT	Gets the value
Set [▶ 1805]	ITcVnAccess_TcVnVector2_INT	Sets the value.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.16.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnVector2_INT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector2_INT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.16.2 Set

Sets the value.

Syntax

Definition:

```
HRESULT Set(
    TcVnVector2_INT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector2_INT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.17 ITcVnAccess_TcVnVector2_REAL

Offers an access interface for TcVnVector2_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector2_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1806]	ITcVnAccess_TcVnVector2_REAL	Gets the value
Set [▶ 1807]	ITcVnAccess_TcVnVector2_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.17.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnVector2_REAL& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector2_REAL [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.17.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    TcVnVector2_REAL& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector2_REAL [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.18 ITcVnAccess_TcVnVector2_SINT

Offers an access interface for TcVnVector2_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector2_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1807]	ITcVnAccess_TcVnVector2_SINT	Gets the value
Set [▶ 1808]	ITcVnAccess_TcVnVector2_SINT	Sets the value.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.18.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnVector2_SINT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector2_SINT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.18.2 Set

Sets the value.

Syntax

Definition:

```
HRESULT Set(
    TcVnVector2_SINT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector2_SINT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.19 ITcVnAccess_TcVnVector2_UINT

Offers an access interface for TcVnVector2_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector2_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1809]	ITcVnAccess_TcVnVector2_UINT	Gets the value
Set [▶ 1809]	ITcVnAccess_TcVnVector2_UINT	Sets the value.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.19.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnVector2_UINT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector2_UINT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.19.2 Set

Sets the value.

Syntax

Definition:

```
HRESULT Set(
    TcVnVector2_UINT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector2_UINT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.20 ITcVnAccess_TcVnVector2_USINT

Offers an access interface for TcVnVector2_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector2_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1810]	ITcVnAccess_TcVnVector2_USINT	Gets the value
Set [▶ 1810]	ITcVnAccess_TcVnVector2_USINT	Sets the value.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.20.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnVector2_USINT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector2_USINT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.20.2 Set

Sets the value.

Syntax

Definition:

```
HRESULT Set(
    TcVnVector2_USINT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector2_USINT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.21 ITcVnAccess_TcVnVector3_INT

Offers an access interface for TcVnVector3_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector3_INT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1811]	ITcVnAccess_TcVnVector3_INT	Gets the value
Set [▶ 1811]	ITcVnAccess_TcVnVector3_INT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.21.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnVector3_INT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector3_INT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.21.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    TcVnVector3_INT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector3_INT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.22 ITcVnAccess_TcVnVector3_REAL

Offers an access interface for TcVnVector3_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector3_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1812]	ITcVnAccess_TcVnVector3_REAL	Gets the value
Set [▶ 1813]	ITcVnAccess_TcVnVector3_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.22.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnVector3_REAL& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector3_REAL [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.22.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    TcVnVector3_REAL& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector3_REAL [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.23 ITcVnAccess_TcVnVector3_SINT

Offers an access interface for TcVnVector3_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector3_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1813]	ITcVnAccess_TcVnVector3_SINT	Gets the value
Set [▶ 1814]	ITcVnAccess_TcVnVector3_SINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.23.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnVector3_SINT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector3_SINT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.23.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    TcVnVector3_SINT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector3_SINT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.24 ITcVnAccess_TcVnVector3_UINT

Offers an access interface for TcVnVector3_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector3_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1815]	ITcVnAccess_TcVnVector3_UINT	Gets the value
Set [▶ 1815]	ITcVnAccess_TcVnVector3_UINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.24.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnVector3_UINT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector3_UINT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.24.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    TcVnVector3_UINT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector3_UINT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.25 ITcVnAccess_TcVnVector3_USINT

Offers an access interface for TcVnVector3_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector3_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1816]	ITcVnAccess_TcVnVector3_USINT	Gets the value
Set [▶ 1816]	ITcVnAccess_TcVnVector3_USINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.25.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnVector3_USINT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector3_USINT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.25.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    TcVnVector3_USINT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector3_USINT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.26 ITcVnAccess_TcVnVector4_DINT

Offers an access interface for TcVnVector4_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector4_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1817]	ITcVnAccess_TcVnVector4_DINT	Gets the value
Set [▶ 1817]	ITcVnAccess_TcVnVector4_DINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.26.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnVector4_DINT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector4_DINT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.26.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    TcVnVector4_DINT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector4_DINT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.27 ITcVnAccess_TcVnVector4_INT

Offers an access interface for TcVnVector4_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector4_INT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1818]	ITcVnAccess_TcVnVector4_INT	Gets the value
Set [▶ 1819]	ITcVnAccess_TcVnVector4_INT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.27.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnVector4_INT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector4_INT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.27.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    TcVnVector4_INT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector4_INT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.28 ITcVnAccess_TcVnVector4_LREAL

Offers an access interface for TcVnVector4_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector4_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1819]	ITcVnAccess_TcVnVector4_LREAL	Gets the value
Set [▶ 1820]	ITcVnAccess_TcVnVector4_LREAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.28.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnVector4_LREAL& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.28.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    TcVnVector4_LREAL& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector4_LREAL [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.29 ITcVnAccess_TcVnVector4_SINT

Offers an access interface for TcVnVector4_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector4_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1821]	ITcVnAccess_TcVnVector4_SINT	Gets the value
Set [▶ 1821]	ITcVnAccess_TcVnVector4_SINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.29.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnVector4_SINT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector4_SINT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.29.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    TcVnVector4_SINT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector4_SINT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.30 ITcVnAccess_TcVnVector4_UINT

Offers an access interface for TcVnVector4_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector4_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1822]	ITcVnAccess_TcVnVector4_UINT	Gets the value
Set [▶ 1822]	ITcVnAccess_TcVnVector4_UINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.30.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnVector4_UINT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector4_UINT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.30.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    TcVnVector4_UINT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector4_UINT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.31 ITcVnAccess_TcVnVector4_USINT

Offers an access interface for TcVnVector4_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector4_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1823]	ITcVnAccess_TcVnVector4_USINT	Gets the value
Set [▶ 1823]	ITcVnAccess_TcVnVector4_USINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.31.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnVector4_USINT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector4_USINT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.31.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    TcVnVector4_USINT& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector4_USINT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.32 ITcVnAccess_UDINT

Offers an access interface for UDINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_UDINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1824]	ITcVnAccess_UDINT	Gets the value
Set [▶ 1825]	ITcVnAccess_UDINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.32.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    ULONG& nValue
)
```

Parameters

Name	Type	Description
nValue	ULONG&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.32.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    ULONG nValue
)
```

Parameters

Name	Type	Description
nValue	ULONG	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.33 ITcVnAccess_UINT

Offers an access interface for UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1825]	ITcVnAccess_UINT	Gets the value
Set [▶ 1826]	ITcVnAccess_UINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.33.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    USHORT& nValue
)
```

Parameters

Name	Type	Description
nValue	USHORT&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.33.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    USHORT nValue
)
```

Parameters

Name	Type	Description
nValue	USHORT	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.34 ITcVnAccess_ULINT

Offers an access interface for ULINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_ULINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1827]	ITcVnAccess_ULINT	Gets the value
Set [▶ 1827]	ITcVnAccess_ULINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.34.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    ULONGLONG& nValue
)
```

Parameters

Name	Type	Description
nValue	ULONGLONG&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.34.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    ULONGLONG nValue
)
```

Parameters

Name	Type	Description
nValue	ULONGLONG	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.35 ITcVnAccess_USINT

Offers an access interface for USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1828]	ITcVnAccess_USINT	Gets the value
Set [▶ 1828]	ITcVnAccess_USINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.35.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    unsigned char& nValue
)
```

Parameters

Name	Type	Description
nValue	unsigned char&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.35.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    unsigned char nValue
)
```

Parameters

Name	Type	Description
nValue	unsigned char	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.36 ITcVnAccess_TcVnVector4_REAL

Offers an access interface for TcVnVector4_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector4_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1829]	ITcVnAccess_TcVnVector4_REAL	Gets the value
Set [▶ 1829]	ITcVnAccess_TcVnVector4_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.36.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnVector4_REAL& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector4_REAL [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.36.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    TcVnVector4_REAL& aVector
)
```

Parameters

Name	Type	Description
aVector	TcVnVector4_REAL [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.2.37 ITcVnAccess_TcVnRotatedRectangle

Offers an access interface for TcVnRotatedRectangle values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnRotatedRectangle

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1830]	ITcVnAccess_TcVnRotatedRectangle	Gets the value
Set [▶ 1830]	ITcVnAccess_TcVnRotatedRectangle	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.2.37.1 Get

Gets the value

Syntax

Definition:

```
HRESULT Get(
    TcVnRotatedRectangle& stRectangle
)
```

Parameters

Name	Type	Description
stRectangle	TcVnRotatedRectangle [▶ 1782]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.2.37.2 Set

Sets the value

Syntax

Definition:

```
HRESULT Set(
    TcVnRotatedRectangle& stRectangle
)
```

Parameters

Name	Type	Description
stRectangle	TcVnRotatedRectangle [▶ 1782]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3 ITcVnRandomAccess

Das Interface `ITcVnRandomAccess` dient dem beliebigen Zugriff auf Elemente eines Containers [▶ 164] und wird von einem ITcVnRandomAccessIterator [▶ 1887] bereitgestellt. Im Gegensatz zu ITcVnAccess [▶ 1787] können beliebige Elemente eines Containers adressiert werden.

6.2.2.1.3.1 ITcVnRandomAccess_DINT

Offers a random access interface for DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_DINT [▶ 1787]
    ITcVnRandomAccess_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1787]	ITcVnAccess_DINT	Gets the value
Set [▶ 1787]	ITcVnAccess_DINT	Sets the value
GetAt [▶ 1831]	ITcVnRandomAccess_DINT	Gets the value
SetAt [▶ 1832]	ITcVnRandomAccess_DINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.1.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG nOffset,
    LONG&    nValue
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
nValue	LONG&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.1.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG nOffset,
    LONG     nValue
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
nValue	LONG	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.2 ITcVnRandomAccess_INT

Offers a random access interface for INT values

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_INT [▶ 1788]
    ITcVnRandomAccess_INT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1788]	ITcVnAccess_INT	Gets the value
Set [▶ 1789]	ITcVnAccess_INT	Sets the value
GetAt [▶ 1833]	ITcVnRandomAccess_INT	Gets the value
SetAt [▶ 1833]	ITcVnRandomAccess_INT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.2.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG nOffset,
    SHORT&   nValue
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
nValue	SHORT&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.2.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG nOffset,
    SHORT    nValue
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
nValue	SHORT	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.3 ITcVnRandomAccess_ITcVnImage

Offers an random access interface for images.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_ITcVnImage [▶ 1789]
    ITcVnRandomAccess_ITcVnImage

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1789]	ITcVnAccess_ITcVnImage	Gets the image.
Set [▶ 1790]	ITcVnAccess_ITcVnImage	Sets the image.
GetAt [▶ 1834]	ITcVnRandomAccess_ITcVnImage	Gets the image.
SetAt [▶ 1834]	ITcVnRandomAccess_ITcVnImage	Sets the image.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.3.1 GetAt

Gets the image.

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG     nOffset,
    ITcVnImage*& ipImage
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position.
ipImage	ITcVnImage* [▶ 1925]&	Returns the image.

Return value

HRESULT [▶ 150]

6.2.2.1.3.3.2 SetAt

Sets the image.

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG    nOffset,
    ITcVnImage* ipImage
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
ipImage	ITcVnImage* [▶ 1925]	The image to set.

Return value

HRESULT [▶ 150]

6.2.2.1.3.4 ITcVnRandomAccess_LREAL

Offers a random access interface for LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_LREAL [▶ 1790]
    ITcVnRandomAccess_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1791]	ITcVnAccess_LREAL	Gets the value
Set [▶ 1791]	ITcVnAccess_LREAL	Sets the value
GetAt [▶ 1835]	ITcVnRandomAccess_LREAL	Gets the value
SetAt [▶ 1836]	ITcVnRandomAccess_LREAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.4.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG nOffset,
    double&  fValue
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
fValue	double&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.4.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG nOffset,
    double   fValue
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position.
fValue	double	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.5 ITcVnRandomAccess_REAL

Offers a random access interface for REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_REAL [▶ 1791]
    ITcVnRandomAccess_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1792]	ITcVnAccess_REAL	Gets the value
Set [▶ 1792]	ITcVnAccess_REAL	Sets the value
GetAt [▶ 1837]	ITcVnRandomAccess_REAL	Gets the value
SetAt [▶ 1837]	ITcVnRandomAccess_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.5.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG nOffset,
    float&   fValue
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
fValue	float&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.5.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG nOffset,
    float    fValue
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
fValue	float	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.6 ITcVnRandomAccess_SINT

Offers a random access interface for SINT values

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_SINT [▶ 1793]
    ITcVnRandomAccess_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1793]	ITcVnAccess_SINT	Gets the value
Set [▶ 1793]	ITcVnAccess_SINT	Sets the value
GetAt [▶ 1838]	ITcVnRandomAccess_SINT	Gets the value
SetAt [▶ 1838]	ITcVnRandomAccess_SINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.6.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG nOffset,
    char&    nValue
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
nValue	char&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.6.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG nOffset,
    char     nValue
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
nValue	char	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.7 ITcVnRandomAccess_TcVnDMatch

Offers an random access interface for TcVnDMatch values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnDMatch [▶ 1794]
    ITcVnRandomAccess_TcVnDMatch

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1794]	ITcVnAccess_TcVnDMatch	Gets the value
Set [▶ 1795]	ITcVnAccess_TcVnDMatch	Sets the value
GetAt [▶ 1839]	ITcVnRandomAccess_TcVnDMatch	Gets the value
SetAt [▶ 1840]	ITcVnRandomAccess_TcVnDMatch	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.7.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG    nOffset,
    TcVnDMatch& stDMatch
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position.
stDMatch	TcVnDMatch [▶ 1768]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.7.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG    nOffset,
    TcVnDMatch& stDMatch
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
stDMatch	TcVnDMatch [▶ 1768]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.8 ITcVnRandomAccess_TcVnKeyPoint

Offers a random access interface for TcVnKeyPoint values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnKeyPoint [▶ 1795]
    ITcVnRandomAccess_TcVnKeyPoint

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1795]	ITcVnAccess_TcVnKeyPoint	Gets the value.
Set [▶ 1796]	ITcVnAccess_TcVnKeyPoint	Sets the value.
GetAt [▶ 1841]	ITcVnRandomAccess_TcVnKeyPoint	Gets the value
SetAt [▶ 1841]	ITcVnRandomAccess_TcVnKeyPoint	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.8.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG      nOffset,
    TcVnKeyPoint& stKeyPoint
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
stKeyPoint	TcVnKeyPoint [▶ 1768]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.8.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG      nOffset,
    TcVnKeyPoint& stKeyPoint
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
stKeyPoint	TcVnKeyPoint [▶ 1768]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.9 ITcVnRandomAccess_TcVnPoint2_DINT

Offers a random access interface for TcVnPoint2_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnPoint2_DINT [▶ 1796]
    ITcVnRandomAccess_TcVnPoint2_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1797]	ITcVnAccess_TcVnPoint2_DINT	Gets the value.
Set [▶ 1797]	ITcVnAccess_TcVnPoint2_DINT	Sets the value.
GetAt [▶ 1842]	ITcVnRandomAccess_TcVnPoint2_DINT	Gets the value
SetAt [▶ 1842]	ITcVnRandomAccess_TcVnPoint2_DINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.9.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG         nOffset,
    TcVnPoint2_DINT& aPoint
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aPoint	TcVnPoint2_DINT [▶ 1711]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.9.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG         nOffset,
    TcVnPoint2_DINT& aPoint
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aPoint	TcVnPoint2_DINT [▶ 1711]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.10 ITcVnRandomAccess_TcVnPoint2_LREAL

Offers a random access interface for TcVnPoint2_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnPoint2_LREAL [▶ 1797]
    ITcVnRandomAccess_TcVnPoint2_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1798]	ITcVnAccess_TcVnPoint2_LREAL	Gets the value
Set [▶ 1798]	ITcVnAccess_TcVnPoint2_LREAL	Sets the value
GetAt [▶ 1843]	ITcVnRandomAccess_TcVnPoint2_LREAL	Gets the value
SetAt [▶ 1844]	ITcVnRandomAccess_TcVnPoint2_LREAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.10.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG          nOffset,
    TcVnPoint2_LREAL& aPoint
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aPoint	TcVnPoint2_LREAL [▶ 1711]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.10.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG          nOffset,
    TcVnPoint2_LREAL& aPoint
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aPoint	TcVnPoint2_LREAL [▶ 1711]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.11 ITcVnRandomAccess_TcVnPoint2_REAL

Offers a random access interface for TcVnPoint2_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnPoint2_REAL [▶ 1799]
    ITcVnRandomAccess_TcVnPoint2_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1799]	ITcVnAccess_TcVnPoint2_REAL	Gets the value
Set [▶ 1799]	ITcVnAccess_TcVnPoint2_REAL	Sets the value
GetAt [▶ 1845]	ITcVnRandomAccess_TcVnPoint2_REAL	Gets the value
SetAt [▶ 1845]	ITcVnRandomAccess_TcVnPoint2_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.11.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG         nOffset,
    TcVnPoint2_REAL& aPoint
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aPoint	TcVnPoint2_REAL [▶ 1711]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.11.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG         nOffset,
    TcVnPoint2_REAL& aPoint
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aPoint	TcVnPoint2_REAL [▶ 1711]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.12 ITcVnRandomAccess_TcVnPoint3_LREAL

Offers a random access interface for TcVnPoint3_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnPoint3_LREAL [▶ 1800]
    ITcVnRandomAccess_TcVnPoint3_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1800]	ITcVnAccess_TcVnPoint3_LREAL	Gets the value
Set [▶ 1801]	ITcVnAccess_TcVnPoint3_LREAL	Sets the value
GetAt [▶ 1846]	ITcVnRandomAccess_TcVnPoint3_LREAL	Gets the value
SetAt [▶ 1846]	ITcVnRandomAccess_TcVnPoint3_LREAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.12.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG          nOffset,
    TcVnPoint3_LREAL& aPoint
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aPoint	TcVnPoint3_LREAL [▶ 1711]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.12.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG          nOffset,
    TcVnPoint3_LREAL& aPoint
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aPoint	TcVnPoint3_LREAL [▶ 1711]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.13 ITcVnRandomAccess_TcVnPoint3_REAL

Offers a random access interface for TcVnPoint3_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnPoint3_REAL [▶ 1801]
    ITcVnRandomAccess_TcVnPoint3_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1801]	ITcVnAccess_TcVnPoint3_REAL	Gets the value
Set [▶ 1802]	ITcVnAccess_TcVnPoint3_REAL	Sets the value
GetAt [▶ 1847]	ITcVnRandomAccess_TcVnPoint3_REAL	Gets the value
SetAt [▶ 1848]	ITcVnRandomAccess_TcVnPoint3_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.13.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG          nOffset,
    TcVnPoint3_REAL& aPoint
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aPoint	TcVnPoint3_REAL [▶ 1711]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.13.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG         nOffset,
    TcVnPoint3_REAL& aPoint
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aPoint	TcVnPoint3_REAL [▶ 1711]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.14 ITcVnRandomAccess_TcVnRectangle_DINT

Offers a random access interface for TcVnRectangle_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnRectangle_DINT [▶ 1802]
    ITcVnRandomAccess_TcVnRectangle_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1803]	ITcVnAccess_TcVnRectangle_DINT	Gets the value
Set [▶ 1803]	ITcVnAccess_TcVnRectangle_DINT	Sets the value
GetAt [▶ 1849]	ITcVnRandomAccess_TcVnRectangle_DINT	Gets the value
SetAt [▶ 1849]	ITcVnRandomAccess_TcVnRectangle_DINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.14.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG            nOffset,
    TcVnRectangle_DINT& stRectangle
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
stRectangle	TcVnRectangle_DINT [▶ 1781]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.14.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG            nOffset,
    TcVnRectangle_DINT& stRectangle
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
stRectangle	TcVnRectangle_DINT [▶ 1781]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.15 ITcVnRandomAccess_TcVnVector2_DINT

Offers a random access interface for TcVnVector2_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector2_DINT [▶ 1803]
    ITcVnRandomAccess_TcVnVector2_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1804]	ITcVnAccess_TcVnVector2_DINT	Gets the value
Set [▶ 1804]	ITcVnAccess_TcVnVector2_DINT	Sets the value.
GetAt [▶ 1850]	ITcVnRandomAccess_TcVnVector2_DINT	Gets the value
SetAt [▶ 1850]	ITcVnRandomAccess_TcVnVector2_DINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.15.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG          nOffset,
    TcVnVector2_DINT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector2_DINT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.15.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG          nOffset,
    TcVnVector2_DINT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector2_DINT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.16 ITcVnRandomAccess_TcVnVector2_INT

Offers a random access interface for TcVnVector2_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector2_INT [▶ 1805]
    ITcVnRandomAccess_TcVnVector2_INT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1805]	ITcVnAccess_TcVnVector2_INT	Gets the value
Set [▶ 1805]	ITcVnAccess_TcVnVector2_INT	Sets the value.
GetAt [▶ 1851]	ITcVnRandomAccess_TcVnVector2_INT	Gets the value
SetAt [▶ 1852]	ITcVnRandomAccess_TcVnVector2_INT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.16.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG          nOffset,
    TcVnVector2_INT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector2_INT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.16.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG         nOffset,
    TcVnVector2_INT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector2_INT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.17 ITcVnRandomAccess_TcVnVector2_REAL

Offers a random access interface for TcVnVector2_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector2_REAL [▶ 1806]
    ITcVnRandomAccess_TcVnVector2_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1806]	ITcVnAccess_TcVnVector2_REAL	Gets the value
Set [▶ 1807]	ITcVnAccess_TcVnVector2_REAL	Sets the value
GetAt [▶ 1853]	ITcVnRandomAccess_TcVnVector2_REAL	Gets the value
SetAt [▶ 1853]	ITcVnRandomAccess_TcVnVector2_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.17.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG          nOffset,
    TcVnVector2_REAL& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector2_REAL [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.17.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG          nOffset,
    TcVnVector2_REAL& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector2_REAL [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.18 ITcVnRandomAccess_TcVnVector2_SINT

Offers a random access interface for TcVnVector2_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector2_SINT [▶ 1807]
    ITcVnRandomAccess_TcVnVector2_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1807]	ITcVnAccess_TcVnVector2_SINT	Gets the value
Set [▶ 1808]	ITcVnAccess_TcVnVector2_SINT	Sets the value.
GetAt [▶ 1854]	ITcVnRandomAccess_TcVnVector2_SINT	Gets the value
SetAt [▶ 1854]	ITcVnRandomAccess_TcVnVector2_SINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.18.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG          nOffset,
    TcVnVector2_SINT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector2_SINT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.18.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG          nOffset,
    TcVnVector2_SINT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector2_SINT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.19 ITcVnRandomAccess_TcVnVector2_UINT

Offers a random access interface for TcVnVector2_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector2_UINT [▶ 1808]
    ITcVnRandomAccess_TcVnVector2_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1809]	ITcVnAccess_TcVnVector2_UINT	Gets the value
Set [▶ 1809]	ITcVnAccess_TcVnVector2_UINT	Sets the value.
GetAt [▶ 1855]	ITcVnRandomAccess_TcVnVector2_UINT	Gets the value
SetAt [▶ 1856]	ITcVnRandomAccess_TcVnVector2_UINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.19.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG          nOffset,
    TcVnVector2_UINT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector2_UINT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.19.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG          nOffset,
    TcVnVector2_UINT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector2_UINT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.20 ITcVnRandomAccess_TcVnVector2_USINT

Offers a random access interface for TcVnVector2_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector2_USINT [▶ 1809]
    ITcVnRandomAccess_TcVnVector2_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1810]	ITcVnAccess_TcVnVector2_USINT	Gets the value
Set [▶ 1810]	ITcVnAccess_TcVnVector2_USINT	Sets the value.
GetAt [▶ 1857]	ITcVnRandomAccess_TcVnVector2_USINT	Gets the value
SetAt [▶ 1857]	ITcVnRandomAccess_TcVnVector2_USINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.20.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG           nOffset,
    TcVnVector2_USINT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector2_USINT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.20.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG           nOffset,
    TcVnVector2_USINT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector2_USINT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.21 ITcVnRandomAccess_TcVnVector3_INT

Offers a random access interface for TcVnVector3_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector3_INT [▶ 1811]
    ITcVnRandomAccess_TcVnVector3_INT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1811]	ITcVnAccess_TcVnVector3_INT	Gets the value
Set [▶ 1811]	ITcVnAccess_TcVnVector3_INT	Sets the value
GetAt [▶ 1858]	ITcVnRandomAccess_TcVnVector3_INT	Gets the value
SetAt [▶ 1858]	ITcVnRandomAccess_TcVnVector3_INT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.21.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG         nOffset,
    TcVnVector3_INT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector3_INT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.21.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG         nOffset,
    TcVnVector3_INT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector3_INT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.22 ITcVnRandomAccess_TcVnVector3_REAL

Offers a random access interface for TcVnVector3_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector3_REAL [▶ 1812]
    ITcVnRandomAccess_TcVnVector3_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1812]	ITcVnAccess_TcVnVector3_REAL	Gets the value
Set [▶ 1813]	ITcVnAccess_TcVnVector3_REAL	Sets the value
GetAt [▶ 1859]	ITcVnRandomAccess_TcVnVector3_REAL	Gets the value
SetAt [▶ 1860]	ITcVnRandomAccess_TcVnVector3_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.22.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG          nOffset,
    TcVnVector3_REAL& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector3_REAL [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.22.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG          nOffset,
    TcVnVector3_REAL& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector3_REAL [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.23 ITcVnRandomAccess_TcVnVector3_SINT

Offers a random access interface for TcVnVector3_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector3_SINT [▶ 1813]
    ITcVnRandomAccess_TcVnVector3_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1813]	ITcVnAccess_TcVnVector3_SINT	Gets the value
Set [▶ 1814]	ITcVnAccess_TcVnVector3_SINT	Sets the value
GetAt [▶ 1861]	ITcVnRandomAccess_TcVnVector3_SINT	Gets the value
SetAt [▶ 1861]	ITcVnRandomAccess_TcVnVector3_SINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.23.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG          nOffset,
    TcVnVector3_SINT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector3_SINT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.23.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG          nOffset,
    TcVnVector3_SINT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector3_SINT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.24 ITcVnRandomAccess_TcVnVector3_UINT

Offers a random access interface for TcVnVector3_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector3_UINT [▶ 1814]
    ITcVnRandomAccess_TcVnVector3_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1815]	ITcVnAccess_TcVnVector3_UINT	Gets the value
Set [▶ 1815]	ITcVnAccess_TcVnVector3_UINT	Sets the value
GetAt [▶ 1862]	ITcVnRandomAccess_TcVnVector3_UINT	Gets the value
SetAt [▶ 1862]	ITcVnRandomAccess_TcVnVector3_UINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.24.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG          nOffset,
    TcVnVector3_UINT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector3_UINT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.24.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG          nOffset,
    TcVnVector3_UINT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector3_UINT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.25 ITcVnRandomAccess_TcVnVector3_USINT

Offers a random access interface for TcVnVector3_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector3_USINT [▶ 1815]
    ITcVnRandomAccess_TcVnVector3_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1816]	ITcVnAccess_TcVnVector3_USINT	Gets the value
Set [▶ 1816]	ITcVnAccess_TcVnVector3_USINT	Sets the value
GetAt [▶ 1863]	ITcVnRandomAccess_TcVnVector3_USINT	Gets the value
SetAt [▶ 1864]	ITcVnRandomAccess_TcVnVector3_USINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.25.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG          nOffset,
    TcVnVector3_USINT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector3_USINT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.25.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG           nOffset,
    TcVnVector3_USINT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector3_USINT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.26 ITcVnRandomAccess_TcVnVector4_DINT

Offers a random access interface for TcVnVector4_DINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector4_DINT [▶ 1817]
    ITcVnRandomAccess_TcVnVector4_DINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1817]	ITcVnAccess_TcVnVector4_DINT	Gets the value
Set [▶ 1817]	ITcVnAccess_TcVnVector4_DINT	Sets the value
GetAt [▶ 1865]	ITcVnRandomAccess_TcVnVector4_DINT	Gets the value
SetAt [▶ 1865]	ITcVnRandomAccess_TcVnVector4_DINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.26.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG          nOffset,
    TcVnVector4_DINT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector4_DINT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.26.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG          nOffset,
    TcVnVector4_DINT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector4_DINT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.27 ITcVnRandomAccess_TcVnVector4_INT

Offers a random access interface for TcVnVector4_INT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector4_INT [▶ 1818]
    ITcVnRandomAccess_TcVnVector4_INT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1818]	ITcVnAccess_TcVnVector4_INT	Gets the value
Set [▶ 1819]	ITcVnAccess_TcVnVector4_INT	Sets the value
GetAt [▶ 1866]	ITcVnRandomAccess_TcVnVector4_INT	Gets the value
SetAt [▶ 1866]	ITcVnRandomAccess_TcVnVector4_INT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.27.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG         nOffset,
    TcVnVector4_INT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector4_INT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.27.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG         nOffset,
    TcVnVector4_INT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector4_INT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.28 ITcVnRandomAccess_TcVnVector4_LREAL

Offers a random access interface for TcVnVector4_LREAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector4_LREAL [▶ 1819]
    ITcVnRandomAccess_TcVnVector4_LREAL

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1819]	ITcVnAccess_TcVnVector4_LREAL	Gets the value
Set [▶ 1820]	ITcVnAccess_TcVnVector4_LREAL	Sets the value
GetAt [▶ 1867]	ITcVnRandomAccess_TcVnVector4_LREAL	Gets the value
SetAt [▶ 1868]	ITcVnRandomAccess_TcVnVector4_LREAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.28.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG           nOffset,
    TcVnVector4_LREAL& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector4_LREAL [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.28.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG           nOffset,
    TcVnVector4_LREAL& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector4_LREAL [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.29 ITcVnRandomAccess_TcVnVector4_SINT

Offers a random access interface for TcVnVector4_SINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector4_SINT [▶ 1820]
    ITcVnRandomAccess_TcVnVector4_SINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1821]	ITcVnAccess_TcVnVector4_SINT	Gets the value
Set [▶ 1821]	ITcVnAccess_TcVnVector4_SINT	Sets the value
GetAt [▶ 1869]	ITcVnRandomAccess_TcVnVector4_SINT	Gets the value
SetAt [▶ 1869]	ITcVnRandomAccess_TcVnVector4_SINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.29.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG          nOffset,
    TcVnVector4_SINT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector4_SINT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.29.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG          nOffset,
    TcVnVector4_SINT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector4_SINT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.30 ITcVnRandomAccess_TcVnVector4_UINT

Offers a random access interface for TcVnVector4_UINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector4_UINT [▶ 1821]
    ITcVnRandomAccess_TcVnVector4_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1822]	ITcVnAccess_TcVnVector4_UINT	Gets the value
Set [▶ 1822]	ITcVnAccess_TcVnVector4_UINT	Sets the value
GetAt [▶ 1870]	ITcVnRandomAccess_TcVnVector4_UINT	Gets the value
SetAt [▶ 1870]	ITcVnRandomAccess_TcVnVector4_UINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.30.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG          nOffset,
    TcVnVector4_UINT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector4_UINT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.30.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG          nOffset,
    TcVnVector4_UINT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector4_UINT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.31 ITcVnRandomAccess_TcVnVector4_USINT

Offers a random access interface for TcVnVector4_USINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector4_USINT [▶ 1823]
    ITcVnRandomAccess_TcVnVector4_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1823]	ITcVnAccess_TcVnVector4_USINT	Gets the value
Set [▶ 1823]	ITcVnAccess_TcVnVector4_USINT	Sets the value
GetAt [▶ 1871]	ITcVnRandomAccess_TcVnVector4_USINT	Gets the value
SetAt [▶ 1872]	ITcVnRandomAccess_TcVnVector4_USINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.31.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG          nOffset,
    TcVnVector4_USINT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector4_USINT [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.31.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG           nOffset,
    TcVnVector4_USINT& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector4_USINT [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.32 ITcVnRandomAccess_UDINT

Offers a random access interface for UDINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_UDINT [▶ 1824]
    ITcVnRandomAccess_UDINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1824]	ITcVnAccess_UDINT	Gets the value
Set [▶ 1825]	ITcVnAccess_UDINT	Sets the value
GetAt [▶ 1873]	ITcVnRandomAccess_UDINT	Gets the value
SetAt [▶ 1873]	ITcVnRandomAccess_UDINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.32.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG nOffset,
    ULONG&   nValue
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
nValue	ULONG&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.32.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG nOffset,
    ULONG    nValue
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
nValue	ULONG	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.33 ITcVnRandomAccess_UINT

Offers a random access interface for UINT values

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_UINT [▶ 1825]
    ITcVnRandomAccess_UINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1825]	ITcVnAccess_UINT	Gets the value
Set [▶ 1826]	ITcVnAccess_UINT	Sets the value
GetAt [▶ 1874]	ITcVnRandomAccess_UINT	Gets the value
SetAt [▶ 1874]	ITcVnRandomAccess_UINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.33.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG nOffset,
    USHORT&  nValue
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
nValue	USHORT&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.33.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG nOffset,
    USHORT   nValue
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
nValue	USHORT	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.34 ITcVnRandomAccess_ULINT

Offers a randowm access interface for ULINT values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_ULINT [▶ 1826]
    ITcVnRandomAccess_ULINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1827]	ITcVnAccess_ULINT	Gets the value
Set [▶ 1827]	ITcVnAccess_ULINT	Sets the value
GetAt [▶ 1875]	ITcVnRandomAccess_ULINT	Gets the value
SetAt [▶ 1876]	ITcVnRandomAccess_ULINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.34.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG   nOffset,
    ULONGLONG& nValue
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
nValue	ULONGLONG&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.34.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG  nOffset,
    ULONGLONG nValue
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
nValue	ULONGLONG	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.35 ITcVnRandomAccess_USINT

Offers a random access interface for USINT values

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_USINT [▶ 1827]
    ITcVnRandomAccess_USINT

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1828]	ITcVnAccess_USINT	Gets the value
Set [▶ 1828]	ITcVnAccess_USINT	Sets the value
GetAt [▶ 1877]	ITcVnRandomAccess_USINT	Gets the value
SetAt [▶ 1877]	ITcVnRandomAccess_USINT	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.35.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG       nOffset,
    unsigned char& nValue
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
nValue	unsigned char&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.35.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG      nOffset,
    unsigned char nValue
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
nValue	unsigned char	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.36 ITcVnRandomAccess_TcVnVector4_REAL

Offers a random access interface for TcVnVector4_REAL values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnVector4_REAL [▶ 1828]
    ITcVnRandomAccess_TcVnVector4_REAL

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1829]	ITcVnAccess_TcVnVector4_REAL	Gets the value
Set [▶ 1829]	ITcVnAccess_TcVnVector4_REAL	Sets the value
GetAt [▶ 1878]	ITcVnRandomAccess_TcVnVector4_REAL	Gets the value
SetAt [▶ 1878]	ITcVnRandomAccess_TcVnVector4_REAL	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.36.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG          nOffset,
    TcVnVector4_REAL& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector4_REAL [▶ 1713]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.36.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG          nOffset,
    TcVnVector4_REAL& aVector
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
aVector	TcVnVector4_REAL [▶ 1713]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.3.37 ITcVnRandomAccess_TcVnRotatedRectangle

Offers a random access interface for TcVnRotatedRectangle values.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnAccess_TcVnRotatedRectangle [▶ 1830]
    ITcVnRandomAccess_TcVnRotatedRectangle

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Get [▶ 1830]	ITcVnAccess_TcVnRotatedRectangle	Gets the value
Set [▶ 1830]	ITcVnAccess_TcVnRotatedRectangle	Sets the value
GetAt [▶ 1879]	ITcVnRandomAccess_TcVnRotatedRectangle	Gets the value
SetAt [▶ 1880]	ITcVnRandomAccess_TcVnRotatedRectangle	Sets the value

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.3.37.1 GetAt

Gets the value

Syntax

Definition:

```
HRESULT GetAt(
    LONGLONG              nOffset,
    TcVnRotatedRectangle& stRectangle
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
stRectangle	TcVnRotatedRectangle [▶ 1782]&	Returns the value

Return value

HRESULT [▶ 150]

6.2.2.1.3.37.2 SetAt

Sets the value

Syntax

Definition:

```
HRESULT SetAt(
    LONGLONG              nOffset,
    TcVnRotatedRectangle& stRectangle
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset to the current position
stRectangle	TcVnRotatedRectangle [▶ 1782]&	The value to set

Return value

HRESULT [▶ 150]

6.2.2.1.4 Iterators

Diese Gruppe enthält Interfaces zur Handhabung von Iteratoren, um auf Elemente von Containern [▶ 163] zuzugreifen.

6.2.2.1.4.1 ITcVnBidirectionalIterator

Offers an interface for a bidirectional iterator.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnIteratorBase [▶ 1884]
    ITcVnForwardIterator [▶ 1881]
      ITcVnBidirectionalIterator

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
CheckIfBegin [▶ 1884]	ITcVnIteratorBase	Checks if the iterator points to the first element.
CheckIfEnd [▶ 1884]	ITcVnIteratorBase	Checks if the iterator points to the past-the-end element (alternatively use F_VN_CheckIfIteratorIsAtEnd).
GetValueSize [▶ 1885]	ITcVnIteratorBase	Gets the memory size (in byte) required by the value the iterator points to.
GetValueTypeGuid [▶ 1885]	ITcVnIteratorBase	Gets the type GUID of the value the iterator points to.
SetToBegin [▶ 1885]	ITcVnIteratorBase	Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).
SetToEnd [▶ 1886]	ITcVnIteratorBase	Sets the iterator to the past-the-end element.
CheckIfEqualTo [▶ 1882]	ITcVnForwardIterator	Checks if iterator is equal to another iterator.
GetContainer [▶ 1883]	ITcVnForwardIterator	Gets a pointer to the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). (Alternatively use F_VN_GetContainer.)
Increment [▶ 1883]	ITcVnForwardIterator	Increments the iterator. (Alternatively use F_VN_IncrementIterator.)
SetContainer [▶ 1883]	ITcVnForwardIterator	Sets the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use F_VN_SetContainer.)
Decrement [▶ 1881]	ITcVnBidirectionalIterator	Decrements the iterator.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.4.1.1 Decrement

Decrements the iterator.

Syntax

Definition:

```
HRESULT Decrement()
```

Return value

HRESULT [▶ 150]

6.2.2.1.4.2 ITcVnForwardIterator

Offers an interface for a forward iterator.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnIteratorBase [▶ 1884]
    ITcVnForwardIterator

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
CheckIfBegin [▶ 1884]	ITcVnIteratorBase	Checks if the iterator points to the first element.
CheckIfEnd [▶ 1884]	ITcVnIteratorBase	Checks if the iterator points to the past-the-end element (alternatively use F_VN_CheckIfIteratorIsAtEnd).
GetValueSize [▶ 1885]	ITcVnIteratorBase	Gets the memory size (in byte) required by the value the iterator points to.
GetValueTypeGuid [▶ 1885]	ITcVnIteratorBase	Gets the type GUID of the value the iterator points to.
SetToBegin [▶ 1885]	ITcVnIteratorBase	Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).
SetToEnd [▶ 1886]	ITcVnIteratorBase	Sets the iterator to the past-the-end element.
CheckIfEqualTo [▶ 1882]	ITcVnForwardIterator	Checks if iterator is equal to another iterator.
GetContainer [▶ 1883]	ITcVnForwardIterator	Gets a pointer to the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). (Alternatively use F_VN_GetContainer.)
Increment [▶ 1883]	ITcVnForwardIterator	Increments the iterator. (Alternatively use F_VN_IncrementIterator.)
SetContainer [▶ 1883]	ITcVnForwardIterator	Sets the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use F_VN_SetContainer.)

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.4.2.1 CheckIfEqualTo

Checks if iterator is equal to another iterator.

Syntax

Definition:

```
HRESULT CheckIfEqualTo(
    ITcVnForwardIterator* ipIterator
)
```

Parameters

Name	Type	Description
ipIterator	ITcVnForwardIterator* [▶ 1881]	Iterator interface to compare with

Return value

HRESULT [▶ 150]

6.2.2.1.4.2.2 GetContainer

Gets a pointer to the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). (Alternatively use F_VN_GetContainer.)

Syntax

Definition:

```
HRESULT GetContainer(
    ITcVnContainer** pipContainer
)
```

Parameters

Name	Type	Description
pipContainer	ITcVnContainer* [▶ 1890]*	Returns the container interface.

Return value

HRESULT [▶ 150]

6.2.2.1.4.2.3 Increment

Increments the iterator. (Alternatively use F_VN_IncrementIterator.)

Syntax

Definition:

```
HRESULT Increment()
```

Return value

HRESULT [▶ 150]

6.2.2.1.4.2.4 SetContainer

Sets the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use F_VN_SetContainer.)

Syntax

Definition:

```
HRESULT SetContainer(
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
ipContainer	ITcVnContainer* [▶ 1890]	Container interface of which the content is to be assigned to the current element.

Return value

HRESULT [▶ 150]

6.2.2.1.4.3 ITcVnIteratorBase

Offers a base interface for iterators

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnIteratorBase

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
CheckIfBegin [▶ 1884]	ITcVnIteratorBase	Checks if the iterator points to the first element.
CheckIfEnd [▶ 1884]	ITcVnIteratorBase	Checks if the iterator points to the past-the-end element (alternatively use F_VN_CheckIfIteratorIsAtEnd).
GetValueSize [▶ 1885]	ITcVnIteratorBase	Gets the memory size (in byte) required by the value the iterator points to.
GetValueTypeGuid [▶ 1885]	ITcVnIteratorBase	Gets the type GUID of the value the iterator points to.
SetToBegin [▶ 1885]	ITcVnIteratorBase	Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).
SetToEnd [▶ 1886]	ITcVnIteratorBase	Sets the iterator to the past-the-end element.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.4.3.1 CheckIfBegin

Checks if the iterator points to the first element.

Syntax

Definition:

```
HRESULT CheckIfBegin()
```

Return value

HRESULT [▶ 150]

6.2.2.1.4.3.2 CheckIfEnd

Checks if the iterator points to the past-the-end element (alternatively use F_VN_CheckIfIteratorIsAtEnd).

Syntax

Definition:

```
HRESULT CheckIfEnd()
```

Return value

HRESULT [▶ 150]

6.2.2.1.4.3.3 GetValueSize

Gets the memory size (in byte) required by the value the iterator points to.

Syntax

Definition:

```
HRESULT GetValueSize(
    ULONGLONG& nSize
)
```

Parameters

Name	Type	Description
nSize	ULONGLONG&	Returns the element size in bytes.

Return value

HRESULT [▶ 150]

6.2.2.1.4.3.4 GetValueTypeGuid

Gets the type GUID of the value the iterator points to.

Syntax

Definition:

```
HRESULT GetValueTypeGuid(
    GUID& nTypeGuid
)
```

Parameters

Name	Type	Description
nTypeGuid	GUID&	Returns the type GUID.

Return value

HRESULT [▶ 150]

6.2.2.1.4.3.5 SetToBegin

Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).

Syntax

Definition:

```
HRESULT SetToBegin()
```

Return value

HRESULT [▶ 150]

6.2.2.1.4.3.6 SetToEnd

Sets the iterator to the past-the-end element.

Syntax

Definition:

```
HRESULT SetToEnd()
```

Return value

HRESULT [▶ 150]

6.2.2.1.4.4 ITcVnIteratorCopyCreator

Offers an interface providing a method for creating a new iterator to the same position.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnIteratorCopyCreator

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
Create [▶ 1886]	ITcVnIteratorCopyCreator	Create a new iterator pointing to the position of the calling iterator.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.4.4.1 Create

Create a new iterator pointing to the position of the calling iterator.

Syntax

Definition:

```
HRESULT Create(
    ITcVnForwardIterator** pipIterator
)
```

Parameters

Name	Type	Description
pipIterator	ITcVnForwardIterator* [▶ 1881]*	Returns the created iterator.

Return value

HRESULT [▶ 150]

6.2.2.1.4.5 ITcVnRandomAccessIterator

Offers an interface for a random access iterator.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnIteratorBase [▶ 1884]
    ITcVnForwardIterator [▶ 1881]
      ITcVnBidirectionalIterator [▶ 1880]
        ITcVnRandomAccessIterator

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
CheckIfBegin [▶ 1884]	ITcVnIteratorBase	Checks if the iterator points to the first element.
CheckIfEnd [▶ 1884]	ITcVnIteratorBase	Checks if the iterator points to the past-the-end element (alternatively use F_VN_CheckIfIteratorIsAtEnd).
GetValueSize [▶ 1885]	ITcVnIteratorBase	Gets the memory size (in byte) required by the value the iterator points to.
GetValueTypeGuid [▶ 1885]	ITcVnIteratorBase	Gets the type GUID of the value the iterator points to.
SetToBegin [▶ 1885]	ITcVnIteratorBase	Sets the iterator to the first element (alternatively use F_VN_SetIteratorToBegin).
SetToEnd [▶ 1886]	ITcVnIteratorBase	Sets the iterator to the past-the-end element.
CheckIfEqualTo [▶ 1882]	ITcVnForwardIterator	Checks if iterator is equal to another iterator.
GetContainer [▶ 1883]	ITcVnForwardIterator	Gets a pointer to the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). (Alternatively use F_VN_GetContainer.)
Increment [▶ 1883]	ITcVnForwardIterator	Increments the iterator. (Alternatively use F_VN_IncrementIterator.)
SetContainer [▶ 1883]	ITcVnForwardIterator	Sets the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use F_VN_SetContainer.)
Decrement [▶ 1881]	ITcVnBidirectionalIterator	Decrements the iterator.
CheckIfGreaterThan [▶ 1888]	ITcVnRandomAccessIterator	Checks if the iterator is greater than another iterator.
CheckIfLessThan [▶ 1889]	ITcVnRandomAccessIterator	Checks if the iterator is less than another iterator.
GetContainerAt [▶ 1889]	ITcVnRandomAccessIterator	Gets a pointer to the element at a specific offset from the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). (Alternatively use F_VN_GetAt_ITcVnContainer.)
SetContainerAt [▶ 1890]	ITcVnRandomAccessIterator	Sets the element at a specific offset from the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use F_VN_SetAt_ITcVnContainer.)

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.4.5.1 CheckIfGreaterThan

Checks if the iterator is greater than another iterator.

Syntax

Definition:

```
HRESULT CheckIfGreaterThan(
    ITcVnRandomAccessIterator* ipIterator
)
```

Parameters

Name	Type	Description
ipIterator	ITcVnRandomAccessIterator* [▶ 1887]	Iterator interface to compare with.

Return value

HRESULT [▶ 150]

6.2.2.1.4.5.2 CheckIfLessThan

Checks if the iterator is less than another iterator.

Syntax

Definition:

```
HRESULT CheckIfLessThan(
    ITcVnRandomAccessIterator* ipIterator
)
```

Parameters

Name	Type	Description
ipIterator	ITcVnRandomAccessIterator* [▶ 1887]	Iterator interface to compare with.

Return value

HRESULT [▶ 150]

6.2.2.1.4.5.3 GetContainerAt

Gets a pointer to the element at a specific offset from the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). (Alternatively use F_VN_GetAt_ITcVnContainer.)

Syntax

Definition:

```
HRESULT GetContainerAt(
    LONGLONG          nOffset,
    ITcVnContainer** pipContainer
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset from the current element.
pipContainer	ITcVnContainer* [▶ 1890]*	Returns the pointer to the container interface.

Return value

HRESULT [▶ 150]

6.2.2.1.4.5.4 SetContainerAt

Sets the element at a specific offset from the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use F_VN_SetAt_ITcVnContainer.)

Syntax

Definition:

```
HRESULT SetContainerAt(
    LONGLONG        nOffset,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
nOffset	LONGLONG	Offset from the current element.
ipContainer	ITcVnContainer* [▶ 1890]	Container interface of which the content is to be assigned to the current element.

Return value

HRESULT [▶ 150]

6.2.2.1.5 ITcVnContainer

Offers an interface for an object container.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnContainer

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
CheckIfBasicContainer [▶ 1891]	ITcVnContainer	Checks if the container contains only basic (non-container) elements.
CheckIfEmpty [▶ 1891]	ITcVnContainer	Checks if the container is empty. (Alternatively use F_VN_CheckIfEmpty.)
GetBidirectionalIterator [▶ 1892]	ITcVnContainer	Gets an interface pointer to a bidirectional iterator (if this iterator type is supported by the container) and increment its reference counter.
GetElementNum [▶ 1892]	ITcVnContainer	Gets the size (number of elements) of the container. (Alternatively use F_VN_GetNumberOfElements.)
GetElementSize [▶ 1892]	ITcVnContainer	Gets the size (in byte) of each element in the container.
GetElementTypeGuid [▶ 1893]	ITcVnContainer	Gets the GUID of the container elements.
GetExportSize [▶ 1893]	ITcVnContainer	Gets combined size (in byte) of all elements in the container.
GetForwardIterator [▶ 1894]	ITcVnContainer	Gets an interface pointer to a forward iterator (if this iterator type is supported by the container) and increment its reference counter. (Alternatively use F_VN_GetForwardIterator.)
GetRandomAccessIterator [▶ 1894]	ITcVnContainer	Gets an interface pointer to a random access iterator (if this iterator type is supported by the container) and increment its reference counter. (Alternatively use F_VN_GetRandomAccessIterator.)
GetTypeGuid [▶ 1894]	ITcVnContainer	Gets GUID of the container.
GetTypeName [▶ 1895]	ITcVnContainer	Gets the container type name as a string.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.1.5.1 CheckIfBasicContainer

Checks if the container contains only basic (non-container) elements.

Syntax

Definition:

```
HRESULT CheckIfBasicContainer()
```

Return value

HRESULT [▶ 150]

6.2.2.1.5.2 CheckIfEmpty

Checks if the container is empty. (Alternatively use F_VN_CheckIfEmpty.)

Syntax

Definition:

```
HRESULT CheckIfEmpty()
```

Return value

HRESULT [▶ 150]

6.2.2.1.5.3 GetBidirectionalIterator

Gets an interface pointer to a bidirectional iterator (if this iterator type is supported by the container) and increment its reference counter.

Syntax

Definition:

```
HRESULT GetBidirectionalIterator(
    ITcVnBidirectionalIterator** pipIterator
)
```

Parameters

Name	Type	Description
pipIterator	ITcVnBidirectionalIterator* [▶ 1880]*	Returns the iterator interface.

Return value

HRESULT [▶ 150]

6.2.2.1.5.4 GetElementNum

Gets the size (number of elements) of the container. (Alternatively use F_VN_GetNumberOfElements.)

Syntax

Definition:

```
HRESULT GetElementNum(
    ULONGLONG& nElementNum
)
```

Parameters

Name	Type	Description
nElementNum	ULONGLONG&	Returns the number of elements in the container.

Return value

HRESULT [▶ 150]

6.2.2.1.5.5 GetElementSize

Gets the size (in byte) of each element in the container.

Syntax

Definition:

```
HRESULT GetElementSize(
    ULONGLONG& nSize
)
```

Parameters

Name	Type	Description
nSize	ULONGLONG&	Returns the size in byte of a single element in the container.

Return value

HRESULT [▶ 150]

6.2.2.1.5.6 GetElementTypeGuid

Gets the GUID of the container elements.

Syntax

Definition:

```
HRESULT GetElementTypeGuid(
    GUID& nTypeGuid
)
```

Parameters

Name	Type	Description
nTypeGuid	GUID&	Returns the GUID of the container elements.

Return value

HRESULT [▶ 150]

6.2.2.1.5.7 GetExportSize

Gets combined size (in byte) of all elements in the container.

Syntax

Definition:

```
HRESULT GetExportSize(
    ULONGLONG& nExportSize
)
```

Parameters

Name	Type	Description
nExportSize	ULONGLONG&	Returns the combined size (in byte) of all elements in the container.

Return value

HRESULT [▶ 150]

6.2.2.1.5.8 GetForwardIterator

Gets an interface pointer to a forward iterator (if this iterator type is supported by the container) and increment its reference counter. (Alternatively use F_VN_GetForwardIterator.)

Syntax

Definition:

```
HRESULT GetForwardIterator(
    ITcVnForwardIterator** pipIterator
)
```

Parameters

Name	Type	Description
pipIterator	ITcVnForwardIterator* [▶ 1881]*	Returns the iterator interface.

Return value

HRESULT [▶ 150]

6.2.2.1.5.9 GetRandomAccessIterator

Gets an interface pointer to a random access iterator (if this iterator type is supported by the container) and increment its reference counter. (Alternatively use F_VN_GetRandomAccessIterator.)

Syntax

Definition:

```
HRESULT GetRandomAccessIterator(
    ITcVnRandomAccessIterator** pipIterator
)
```

Parameters

Name	Type	Description
pipIterator	ITcVnRandomAccessIterator* [▶ 1887]*	Returns the iterator interface.

Return value

HRESULT [▶ 150]

6.2.2.1.5.10 GetTypeGuid

Gets GUID of the container.

Syntax

Definition:

```
HRESULT GetTypeGuid(
    GUID& nTypeGuid
)
```

Parameters

Name	Type	Description
nTypeGuid	GUID&	Returns the GUID of the container.

Return value

HRESULT [▶ 150]

6.2.2.1.5.11 GetTypeName

Gets the container type name as a string.

Syntax

Definition:

```
HRESULT GetTypeName(
    PCHAR  sTypeName,
    USHORT nMaxLen
)
```

Parameters

Name	Type	Description
sTypeName	PCHAR	Returns the container type name as a string.
nMaxLen	USHORT	Maximum string length allowed to be written in sTypeName

Return value

HRESULT [▶ 150]

6.2.2.2 Images

Diese Gruppe enthält Interfaces zur Aufnahme und Handhabung von Bildern [▶ 159].

6.2.2.2.1 Acquisition

Diese Gruppe enthält Interfaces zur Handhabung der Bildaufnahme.

6.2.2.2.1.1 ITcIoFileImageAcquisition

Interface for file image acquisition.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcIoImageAcquisition [▶ 1899]
    ITcIoFileImageAcquisition

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
RegisterReceiver [▶ 1900]	ITcIoImageAcquisition	Register callback interface.
UnregisterReceiver [▶ 1900]	ITcIoImageAcquisition	Unregister callback interface.
OpenCamera [▶ 1901]	ITcIoImageAcquisition	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 1901]	ITcIoImageAcquisition	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 1901]	ITcIoImageAcquisition	Start the image acquisition.
StopAcquisition [▶ 1901]	ITcIoImageAcquisition	Stop the image acquisition.
SoftwareTrigger [▶ 1902]	ITcIoImageAcquisition	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 1902]	ITcIoImageAcquisition	Send initialization commands to the camera.
CheckConnection [▶ 1896]	ITcIoFileImageAcquisition	Checks the connection
TriggerImage [▶ 1896]	ITcIoFileImageAcquisition	Initialize the software trigger and trigger a single image. Skips nSkipImages and triggers the capturing of image nSkipImages+1.
TriggerImageByName [▶ 1897]	ITcIoFileImageAcquisition	Initialize the software trigger and trigger a single image specified by its name in the client assistant.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.2.1.1.1 CheckConnection

Checks the connection

Syntax

Definition:

```
HRESULT CheckConnection()
```

Return value

HRESULT [▶ 150]

6.2.2.2.1.1.2 TriggerImage

Initialize the software trigger and trigger a single image. Skips nSkipImages and triggers the capturing of image nSkipImages+1.

Syntax

Definition:

```
HRESULT TriggerImage(
    LONG nSkipImages
)
```

Parameters

Name	Type	Description
nSkipImages	LONG	Number of images to skip

Return value

HRESULT [▶ 150]

6.2.2.2.1.1.3 TriggerImageByName

Initialize the software trigger and trigger a single image specified by its name in the client assistant.

Syntax

Definition:

```
HRESULT TriggerImageByName(
    PCCH sImageName
)
```

Parameters

Name	Type	Description
sImageName	PCCH	Image name to trigger

Return value

HRESULT [▶ 150]

6.2.2.2.1.2 ITcIoFileImageRecv

Interface for a image receiver.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcIoImageRecv [▶ 1902]
    ITcIoFileImageRecv

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
ReceiveImage [▶ 1903]	ITcIoImageRecv	Receive an image from an instance of ITcIoGevImageAcquisition.
ReceiveOpResult [▶ 1903]	ITcIoImageRecv	Receive an operation result from an instance of ITcIoGevImageAcquisition.
ReceiveImage [▶ 1898]	ITcIoFileImageRecv	Receive an image from an instance of ITcIoFileImageAcquisition.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.2.1.2.1 ReceiveImage

Receive an image from an instance of ITcIoFileImageAcquisition.

Syntax

Definition:

```
HRESULT ReceiveImage(
    ITcVnImageBase* ipImage,
    PCHAR           sFileName,
    HRESULT         hrAcquisitionResult
)
```

Parameters

Name	Type	Description
ipImage	ITcVnImageBase* [▶ 1927]	Interface pointer to the image.
sFileName	PCHAR	Returns the filename to the image.
hrAcquisitionResult	HRESULT [▶ 150]	HRESULT indicating the status of the image acquisition.

Return value

HRESULT [▶ 150]

6.2.2.2.1.3 ITcIoGevImageAcquisition

Interface for GigE Vision image acquisition.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcIoImageAcquisition [▶ 1899]
    ITcIoGevImageAcquisition

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
RegisterReceiver [▶ 1900]	ITcIoImageAcquisition	Register callback interface.
UnregisterReceiver [▶ 1900]	ITcIoImageAcquisition	Unregister callback interface.
OpenCamera [▶ 1901]	ITcIoImageAcquisition	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 1901]	ITcIoImageAcquisition	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 1901]	ITcIoImageAcquisition	Start the image acquisition.
StopAcquisition [▶ 1901]	ITcIoImageAcquisition	Stop the image acquisition.
SoftwareTrigger [▶ 1902]	ITcIoImageAcquisition	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 1902]	ITcIoImageAcquisition	Send initialization commands to the camera.
CheckConnection [▶ 1899]	ITcIoGevImageAcquisition	Checks the camera connection

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.2.1.3.1 CheckConnection

Checks the camera connection

Syntax

Definition:

```
HRESULT CheckConnection(
    GEV_CAMERA_STATE& eAssumedState,
    GEV_CAMERA_STATE& eActualState
)
```

Parameters

Name	Type	Description
eAssumedState	GEV_CAMERA_STATE [▶ 1760]&	The internally assumed state of the camera.
eActualState	GEV_CAMERA_STATE [▶ 1760]&	The actually observable state of the camera.

Return value

HRESULT [▶ 150]

6.2.2.2.1.4 ITcIoImageAcquisition

Interface for image acquisition.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcIoImageAcquisition

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
RegisterReceiver [▶ 1900]	ITcIoImageAcquisition	Register callback interface.
UnregisterReceiver [▶ 1900]	ITcIoImageAcquisition	Unregister callback interface.
OpenCamera [▶ 1901]	ITcIoImageAcquisition	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 1901]	ITcIoImageAcquisition	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 1901]	ITcIoImageAcquisition	Start the image acquisition.
StopAcquisition [▶ 1901]	ITcIoImageAcquisition	Stop the image acquisition.
SoftwareTrigger [▶ 1902]	ITcIoImageAcquisition	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 1902]	ITcIoImageAcquisition	Send initialization commands to the camera.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.2.1.4.1 RegisterReceiver

Register callback interface.

Syntax

Definition:

```
HRESULT RegisterReceiver(
    ITcIoImageRecv* ipRecv
)
```

Parameters

Name	Type	Description
ipRecv	ITcIoImageRecv* [▶ 1902]	Pointer to an interface containing the callback function.

Return value

HRESULT [▶ 150]

6.2.2.2.1.4.2 UnregisterReceiver

Unregister callback interface.

Syntax

Definition:

```
HRESULT UnregisterReceiver()
```

Return value

HRESULT [▶ 150]

6.2.2.2.1.4.3 OpenCamera

Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).

Syntax

Definition:

```
HRESULT OpenCamera()
```

Return value

HRESULT [▶ 150]

6.2.2.2.1.4.4 CloseCamera

Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).

Syntax

Definition:

```
HRESULT CloseCamera()
```

Return value

HRESULT [▶ 150]

6.2.2.2.1.4.5 StartAcquisition

Start the image acquisition.

Syntax

Definition:

```
HRESULT StartAcquisition()
```

Return value

HRESULT [▶ 150]

6.2.2.2.1.4.6 StopAcquisition

Stop the image acquisition.

Syntax

Definition:

```
HRESULT StopAcquisition()
```

Return value

HRESULT [▶ 150]

6.2.2.2.1.4.7 SoftwareTrigger

Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.

Syntax

Definition:

```
HRESULT SoftwareTrigger(
    bool bSplitConcatenatedCommands,
    bool bOmitAcknowledgement
)
```

Parameters

Name	Type	Description
bSplitConcatenatedCommands	bool	If true, multi-read and multi-write commands are split into sequences of single read and single write commands, respectively.
bOmitAcknowledgement	bool	Indicates that no acknowledge packet should be requested.

Return value

HRESULT [▶ 150]

6.2.2.2.1.4.8 InitializeCamera

Send initialization commands to the camera.

Syntax

Definition:

```
HRESULT InitializeCamera()
```

Return value

HRESULT [▶ 150]

6.2.2.2.1.5 ITcIoImageRecv

Interface for a image receiver.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcIoImageRecv

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
ReceiveImage [▶ 1903]	ITcIoImageRecv	Receive an image from an instance of ITcIoGevImageAcquisition.
ReceiveOpResult [▶ 1903]	ITcIoImageRecv	Receive an operation result from an instance of ITcIoGevImageAcquisition.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.2.1.5.1 ReceiveImage

Receive an image from an instance of ITcIoGevImageAcquisition.

Syntax

Definition:

```
HRESULT ReceiveImage(
    ITcVnImageBase* ipImage,
    HRESULT         hrAcquisitionResult
)
```

Parameters

Name	Type	Description
ipImage	ITcVnImageBase* [▶ 1927]	Interface pointer to the image.
hrAcquisitionResult	HRESULT [▶ 150]	HRESULT indicating the status of the image acquisition.

Return value

HRESULT [▶ 150]

6.2.2.2.1.5.2 ReceiveOpResult

Receive an operation result from an instance of ITcIoGevImageAcquisition.

Syntax

Definition:

```
HRESULT ReceiveOpResult(
    HRESULT hrOperationResult
)
```

Parameters

Name	Type	Description
hrOperationResult	HRESULT [▶ 150]	HRESULT indicating the status of the performed operation.

Return value

HRESULT [▶ 150]

6.2.2.2.1.6 ITcVnFileImageProvider

Interface for an image provider for images from outside realtime environment.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnImageProvider [▶ 1907]
    ITcVnFileImageProvider

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
CheckIfBusy [▶ 1908]	ITcVnImageProvider	Check whether the module is busy; i.e, an operation is in progress.
CheckIfCameraConnected [▶ 1908]	ITcVnImageProvider	Check if the camera is properly connected (Returns S_OK if connected and S_FALSE if not. Can only check software connection, i.e. the check is unable to determine if the cable is connected or not in case that S_FALSE is returned).
GetCurrentImage [▶ 1909]	ITcVnImageProvider	Gets the current image by detaching its internal reference.
GetError [▶ 1909]	ITcVnImageProvider	Gets result of the last operation performed.
OpenCamera [▶ 1909]	ITcVnImageProvider	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 1910]	ITcVnImageProvider	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 1910]	ITcVnImageProvider	Start image acquisition.
StopAcquisition [▶ 1910]	ITcVnImageProvider	Stop image acquisition.
SoftwareTrigger [▶ 1910]	ITcVnImageProvider	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 1910]	ITcVnImageProvider	Set camera to initial state.
GetCurrentImageAndFileName [▶ 1905]	ITcVnFileImageProvider	Get the current image by detaching its internal reference.
TriggerImage [▶ 1905]	ITcVnFileImageProvider	Initialize the software trigger and trigger a single image. Skips nSkipImages and triggers the capturing of image nSkipImages+1.
TriggerImageByName [▶ 1905]	ITcVnFileImageProvider	Initialize the software trigger and trigger a single image specified by its name in the client assistant.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.2.1.6.1 GetCurrentImageAndFileName

Get the current image by detaching its internal reference.

Syntax

Definition:

```
HRESULT GetCurrentImageAndFileName(
    ITcVnImage** pipImage,
    PCHAR        sFileName,
    USHORT       nMaxLen
)
```

Parameters

Name	Type	Description
pipImage	ITcVnImage* [▶ 1925]*	pointer to the interface pointer to be returned, might be 0 if the image acquisition failed
sFileName	PCHAR	returns the filename as a string
nMaxLen	USHORT	maximum string length allowed to be written in sFileName

Return value

HRESULT [▶ 150]

6.2.2.2.1.6.2 TriggerImage

Initialize the software trigger and trigger a single image. Skips nSkipImages and triggers the capturing of image nSkipImages+1.

Syntax

Definition:

```
HRESULT TriggerImage(
    LONG nSkipImages
)
```

Parameters

Name	Type	Description
nSkipImages	LONG	Number of images to skip

Return value

HRESULT [▶ 150]

6.2.2.2.1.6.3 TriggerImageByName

Initialize the software trigger and trigger a single image specified by its name in the client assistant.

Syntax

Definition:

```
HRESULT TriggerImageByName(
    PCCH sImageName
)
```

Parameters

Name	Type	Description
sImageName	PCCH	Image name to trigger

Return value

HRESULT [▶ 150]

6.2.2.2.1.7 ITcVnGevImageProvider

Interface for a GigE Vision image provider.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnImageProvider [▶ 1907]
    ITcVnGevImageProvider

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
CheckIfBusy [▶ 1908]	ITcVnImageProvider	Check whether the module is busy; i.e, an operation is in progress.
CheckIfCameraConnected [▶ 1908]	ITcVnImageProvider	Check if the camera is properly connected (Returns S_OK if connected and S_FALSE if not. Can only check software connection, i.e. the check is unable to determine if the cable is connected or not in case that S_FALSE is returned).
GetCurrentImage [▶ 1909]	ITcVnImageProvider	Gets the current image by detaching its internal reference.
GetError [▶ 1909]	ITcVnImageProvider	Gets result of the last operation performed.
OpenCamera [▶ 1909]	ITcVnImageProvider	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 1910]	ITcVnImageProvider	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 1910]	ITcVnImageProvider	Start image acquisition.
StopAcquisition [▶ 1910]	ITcVnImageProvider	Stop image acquisition.
SoftwareTrigger [▶ 1910]	ITcVnImageProvider	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 1910]	ITcVnImageProvider	Set camera to initial state.
GetCurrentGevImage [▶ 1907]	ITcVnGevImageProvider	Gets the current GigE Vision image by detaching its internal reference.
GetCurrentImageWithGvspInfo [▶ 1907]	ITcVnGevImageProvider	Gets the current image by detaching its internal reference and additionally provide the GVSP info.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.2.1.7.1 GetCurrentGevImage

Gets the current GigE Vision image by detaching its internal reference.

Syntax

Definition:

```
HRESULT GetCurrentGevImage(
    ITcIoGevImage** pipGevImage
)
```

Parameters

Name	Type	Description
pipGevImage	ITcIoGevImage* [▶ 1922]*	Pointer to the interface pointer to be returned, might be 0 if the image acquisition failed.

Return value

HRESULT [▶ 150]

6.2.2.2.1.7.2 GetCurrentImageWithGvspInfo

Gets the current image by detaching its internal reference and additionally provide the GVSP info.

Syntax

Definition:

```
HRESULT GetCurrentImageWithGvspInfo(
    ITcVnImage**     pipImage,
    GVSP_IMAGE_INFO& stGvspInfo
)
```

Parameters

Name	Type	Description
pipImage	ITcVnImage* [▶ 1925]*	Pointer to the interface pointer to be returned, might be 0 if the image acquisition failed.
stGvspInfo	GVSP_IMAGE_INFO [▶ 1763]&	Contains useful meta information like image id, time stamp etc.

Return value

HRESULT [▶ 150]

6.2.2.2.1.8 ITcVnImageProvider

Interface for an image provider.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnImageProvider

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
CheckIfBusy [▶ 1908]	ITcVnImageProvider	Check whether the module is busy; i.e, an operation is in progress.
CheckIfCameraConnected [▶ 1908]	ITcVnImageProvider	Check if the camera is properly connected (Returns S_OK if connected and S_FALSE if not. Can only check software connection, i.e. the check is unable to determine if the cable is connected or not in case that S_FALSE is returned).
GetCurrentImage [▶ 1909]	ITcVnImageProvider	Gets the current image by detaching its internal reference.
GetError [▶ 1909]	ITcVnImageProvider	Gets result of the last operation performed.
OpenCamera [▶ 1909]	ITcVnImageProvider	Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).
CloseCamera [▶ 1910]	ITcVnImageProvider	Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).
StartAcquisition [▶ 1910]	ITcVnImageProvider	Start image acquisition.
StopAcquisition [▶ 1910]	ITcVnImageProvider	Stop image acquisition.
SoftwareTrigger [▶ 1910]	ITcVnImageProvider	Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.
InitializeCamera [▶ 1910]	ITcVnImageProvider	Set camera to initial state.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.2.1.8.1 CheckIfBusy

Check whether the module is busy; i.e, an operation is in progress.

Syntax

Definition:

```
HRESULT CheckIfBusy()
```

Return value

HRESULT [▶ 150]

6.2.2.2.1.8.2 CheckIfCameraConnected

Check if the camera is properly connected (Returns S_OK if connected and S_FALSE if not. Can only check software connection, i.e. the check is unable to determine if the cable is connected or not in case that S_FALSE is returned).

Syntax

Definition:

```
HRESULT CheckIfCameraConnected()
```

Return value

HRESULT [▶ 150]

6.2.2.2.1.8.3 GetCurrentImage

Gets the current image by detaching its internal reference.

Syntax

Definition:

```
HRESULT GetCurrentImage(
    ITcVnImage** pipImage
)
```

Parameters

Name	Type	Description
pipImage	ITcVnImage* [▶ 1925]*	Pointer to the interface pointer to be returned, might be 0 if the image acquisition failed.

Return value

HRESULT [▶ 150]

6.2.2.2.1.8.4 GetError

Gets result of the last operation performed.

Syntax

Definition:

```
HRESULT GetError()
```

Return value

HRESULT [▶ 150]

6.2.2.2.1.8.5 OpenCamera

Open camera (open the control channel, open a stream channel, and activate the GVSP receiver).

Syntax

Definition:

```
HRESULT OpenCamera()
```

Return value

HRESULT [▶ 150]

6.2.2.2.1.8.6 CloseCamera

Close camera (deactivate the GVSP receiver, close the stream channel, and close the control channel).

Syntax

Definition:

```
HRESULT CloseCamera()
```

Return value

HRESULT [▶ 150]

6.2.2.2.1.8.7 StartAcquisition

Start image acquisition.

Syntax

Definition:

```
HRESULT StartAcquisition()
```

Return value

HRESULT [▶ 150]

6.2.2.2.1.8.8 StopAcquisition

Stop image acquisition.

Syntax

Definition:

```
HRESULT StopAcquisition()
```

Return value

HRESULT [▶ 150]

6.2.2.2.1.8.9 SoftwareTrigger

Initialize the software trigger and trigger it. The initialization step is skipped if it was executed before.

Syntax

Definition:

```
HRESULT SoftwareTrigger()
```

Return value

HRESULT [▶ 150]

6.2.2.2.1.8.10 InitializeCamera

Set camera to initial state.

Syntax

Definition:

```
HRESULT InitializeCamera()
```

Return value

HRESULT [▶ 150]

6.2.2.2.2 Export

Diese Gruppe enthält Interfaces zur Handhabung des Daten-Exports im Zusammenhang mit Bildern.

6.2.2.2.2.1 ITcVnBitmapExport

Interface for exporting an image as a Windows Bitmap (BMP).

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnBitmapExport

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
GetBitmapSize [▶ 1911]	ITcVnBitmapExport	Gets size of the image if it is converted to a Windows Bitmap (BMP).
GetBitmapImage [▶ 1912]	ITcVnBitmapExport	Export the image as a Windows Bitmap (BMP) into a given buffer.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.2.2.1.1 GetBitmapSize

Gets size of the image if it is converted to a Windows Bitmap (BMP).

Syntax

Definition:

```
HRESULT GetBitmapSize(
    ULONGLONG& nBitmapSize,
    ULONG&     nBitmapWidth,
    ULONG&     nBitmapHeight
)
```

Parameters

Name	Type	Description
nBitmapSize	ULONGLONG&	Output parameter containing the required buffer size.
nBitmapWidth	ULONG&	Desired width or 0 to keep the original width (in) and actual width (out).
nBitmapHeight	ULONG&	Desired height or 0 to keep the original height (in) and actual height (out).

Return value

HRESULT [▶ 150]

6.2.2.2.2.1.2 GetBitmapImage

Export the image as a Windows Bitmap (BMP) into a given buffer.

Syntax

Definition:

```
HRESULT GetBitmapImage(
    ULONGLONG& nBitmapSize,
    PVOID      pDestBuffer,
    ULONG&     nBitmapWidth,
    ULONG&     nBitmapHeight
)
```

Parameters

Name	Type	Description
nBitmapSize	ULONGLONG&	Maximum buffer size (in) and actual buffer size (out).
pDestBuffer	PVOID	Pointer to the destination buffer.
nBitmapWidth	ULONG&	Desired width or 0 to keep the original width (in) and actual width (out).
nBitmapHeight	ULONG&	Desired height or 0 to keep the original height (in) and actual height (out).

Return value

HRESULT [▶ 150]

6.2.2.2.2.2 ITcVnBitmapExportNotification

Interface required for sending displayable images as ADS notifications

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnBitmapExportNotification

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
GetTimestamp [▶ 1913]	ITcVnBitmapExportNotification	Gets the timestamp of the latest image change.
GetBitmapImageRpcUnlocked [▶ 1913]	ITcVnBitmapExportNotification	Export the image as a Windows Bitmap (BMP) into a given buffer by means of an unlocked remote procedure call. It lies within the responsibility of the user to ensure that no conflicting accesses can occur.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.2.2.2.1 GetTimestamp

Gets the timestamp of the latest image change.

Syntax

Definition:

```
HRESULT GetTimestamp(
    LONGLONG& nTimestamp
)
```

Parameters

Name	Type	Description
nTimestamp	LONGLONG&	Returns the timestamp.

Return value

HRESULT [▶ 150]

6.2.2.2.2.2.2 GetBitmapImageRpcUnlocked

Export the image as a Windows Bitmap (BMP) into a given buffer by means of an unlocked remote procedure call. It lies within the responsibility of the user to ensure that no conflicting accesses can occur.

Syntax

Definition:

```
HRESULT GetBitmapImageRpcUnlocked(
    ULONGLONG& nBitmapSize,
    PVOID      pDestBuffer,
    ULONG&     nBitmapWidth,
    ULONG&     nBitmapHeight
)
```

Parameters

Name	Type	Description
nBitmapSize	ULONGLONG&	Maximum buffer size (in) and actual buffer size (out).
pDestBuffer	PVOID	Pointer to the destination buffer.
nBitmapWidth	ULONG&	Desired width or 0 to keep the original width (in) and actual width (out).
nBitmapHeight	ULONG&	Desired height or 0 to keep the original height (in) and actual height (out).

Return value

HRESULT [▶ 150]

6.2.2.2.2.3 ITcVnBitmapExportRpcUnlocked

Interface to export an image as a Windows Bitmap (BMP) into a given buffer by means of an unlocked remote procedure call.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnBitmapExport [▶ 1911]
    ITcVnBitmapExportRpcUnlocked

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
GetBitmapSize [▶ 1911]	ITcVnBitmapExport	Gets size of the image if it is converted to a Windows Bitmap (BMP).
GetBitmapImage [▶ 1912]	ITcVnBitmapExport	Export the image as a Windows Bitmap (BMP) into a given buffer.
GetBitmapImageRpcUnlocked [▶ 1914]	ITcVnBitmapExportRpcUnlocked	Export the image as a Windows Bitmap (BMP) into a given buffer by means of an unlocked remote procedure call. It lies within the responsibility of the user to ensure that no conflicting accesses can occur.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.2.2.3.1 GetBitmapImageRpcUnlocked

Export the image as a Windows Bitmap (BMP) into a given buffer by means of an unlocked remote procedure call. It lies within the responsibility of the user to ensure that no conflicting accesses can occur.

Syntax

Definition:

```
HRESULT GetBitmapImageRpcUnlocked(
    ULONGLONG& nBitmapSize,
    PVOID      pDestBuffer,
    ULONG&     nBitmapWidth,
    ULONG&     nBitmapHeight
)
```

Parameters

Name	Type	Description
nBitmapSize	ULONGLONG&	Maximum buffer size (in) and actual buffer size (out).
pDestBuffer	PVOID	Pointer to the destination buffer.
nBitmapWidth	ULONG&	Desired width or 0 to keep the original width (in) and actual width (out).
nBitmapHeight	ULONG&	Desired height or 0 to keep the original height (in) and actual height (out).

Return value

HRESULT [▶ 150]

6.2.2.2.2.4 ITcVnHistogramExport

Interface for exporting a histogram for an image as an array.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnHistogramExport

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
GetHistogramSize [▶ 1916]	ITcVnHistogramExport	Gets the size of the histogram array.
GetHistogramArray [▶ 1916]	ITcVnHistogramExport	Gets the histogram array.
GetCustomHistogramSize [▶ 1916]	ITcVnHistogramExport	Gets the size of a custom histogram array.
GetCustomHistogramArray [▶ 1917]	ITcVnHistogramExport	Gets a custom histogram array.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.2.2.4.1 GetHistogramSize

Gets the size of the histogram array.

Syntax

Definition:

```
HRESULT GetHistogramSize(
    ULONGLONG& nHistogramSize
)
```

Parameters

Name	Type	Description
nHistogramSize	ULONGLONG&	Output parameter containing the required buffer size.

Return value

HRESULT [▶ 150]

6.2.2.2.2.4.2 GetHistogramArray

Gets the histogram array.

Syntax

Definition:

```
HRESULT GetHistogramArray(
    ULONGLONG& nHistogramSize,
    PVOID      pDestBuffer
)
```

Parameters

Name	Type	Description
nHistogramSize	ULONGLONG&	Maximum buffer size (in) and actual buffer size (out).
pDestBuffer	PVOID	Pointer to the destination buffer.

Return value

HRESULT [▶ 150]

6.2.2.2.2.4.3 GetCustomHistogramSize

Gets the size of a custom histogram array.

Syntax

Definition:

```
HRESULT GetCustomHistogramSize(
    ULONGLONG& nHistogramSize,
    ULONG&     nBins
)
```

Parameters

Name	Type	Description
nHistogramSize	ULONGLONG&	Output parameter containing the required buffer size.
nBins	ULONG&	Desired number of bins or 0 to keep the default for the corresponding image format (in) and default number of bins (out).

Return value

HRESULT [▶ 150]

6.2.2.2.2.4.4 GetCustomHistogramArray

Gets a custom histogram array.

Syntax

Definition:

```
HRESULT GetCustomHistogramArray(
    ULONGLONG& nHistogramSize,
    PVOID      pDestBuffer,
    ULONG&     nBins,
    double&    fLowerBound,
    double&    fUpperBound
)
```

Parameters

Name	Type	Description
nHistogramSize	ULONGLONG&	Maximum buffer size (in) and actual buffer size (out).
pDestBuffer	PVOID	Pointer to the destination buffer.
nBins	ULONG&	Desired number of bins or 0 to keep the default for the corresponding image format (in) and default number of bins (out).
fLowerBound	double&	Lower (inclusive) boundary of the 0-th histogram bin (in), or receive the default if fLowerBound AND fUpperBound are set to 0 (out).
fUpperBound	double&	Upper (exclusive) boundary of the last histogram bin nBins-1 (in), or receive the default if fLowerBound AND fUpperBound are set to 0 (out).

Return value

HRESULT [▶ 150]

6.2.2.2.2.5 ITcVnTiffExport

Interface for exporting an image as tiff.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnTiffExport

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
GetTiffSize [▶ 1918]	ITcVnTiffExport	Gets size of the image if it is converted to tiff.
GetTiffImage [▶ 1918]	ITcVnTiffExport	Export the image as tiff into a given buffer.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.2.2.5.1 GetTiffSize

Gets size of the image if it is converted to tiff.

Syntax

Definition:

```
HRESULT GetTiffSize(
    ULONGLONG& nTiffSize
)
```

Parameters

Name	Type	Description
nTiffSize	ULONGLONG&	Output parameter containing the required buffer size.

Return value

HRESULT [▶ 150]

6.2.2.2.2.5.2 GetTiffImage

Export the image as tiff into a given buffer.

Syntax

Definition:

```
HRESULT GetTiffImage(
    ULONGLONG& nTiffSize,
    PVOID      pDestBuffer
)
```

Parameters

Name	Type	Description
nTiffSize	ULONGLONG&	Maximum buffer size (in) and actual buffer size (out).
pDestBuffer	PVOID	Pointer to the destination buffer.

Return value

HRESULT [▶ 150]

6.2.2.2.2.6 ITcVnTiffExportNotification

Interface for sending displayable images as tiff via ADS notifications.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnTiffExportNotification

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
GetTimestamp [▶ 1919]	ITcVnTiffExportNotification	Gets the timestamp of the latest image change.
GetTiffImageRpcUnlocked [▶ 1920]	ITcVnTiffExportNotification	Export the image as tiff into a given buffer by means of an unlocked remote procedure call. It lies within the responsibility of the user to ensure that no conflicting accesses can occur.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.2.2.6.1 GetTimestamp

Gets the timestamp of the latest image change.

Syntax

Definition:

```
HRESULT GetTimestamp(
    LONGLONG& nTimestamp
)
```

Parameters

Name	Type	Description
nTimestamp	LONGLONG&	Returns the timestamp.

Return value

HRESULT [▶ 150]

6.2.2.2.2.6.2 GetTiffImageRpcUnlocked

Export the image as tiff into a given buffer by means of an unlocked remote procedure call. It lies within the responsibility of the user to ensure that no conflicting accesses can occur.

Syntax

Definition:

```
HRESULT GetTiffImageRpcUnlocked(
    ULONGLONG& nTiffSize,
    PVOID      pDestBuffer
)
```

Parameters

Name	Type	Description
nTiffSize	ULONGLONG&	Maximum buffer size (in) and actual buffer size (out).
pDestBuffer	PVOID	Pointer to the destination buffer.

Return value

HRESULT [▶ 150]

6.2.2.2.2.7 ITcVnTiffExportRpcUnlocked

Interface for exporting an image as tiff by means of an unlocked remote procedure call.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnTiffExport [▶ 1917]
    ITcVnTiffExportRpcUnlocked

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
GetTiffSize [▶ 1918]	ITcVnTiffExport	Gets size of the image if it is converted to tiff.
GetTiffImage [▶ 1918]	ITcVnTiffExport	Export the image as tiff into a given buffer.
GetTiffImageRpcUnlocked [▶ 1921]	ITcVnTiffExportRpcUnlocked	Export the image as tiff into a given buffer by means of an unlocked remote procedure call. It lies within the responsibility of the user to ensure that no conflicting accesses can occur.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.2.2.7.1 GetTiffImageRpcUnlocked

Export the image as tiff into a given buffer by means of an unlocked remote procedure call. It lies within the responsibility of the user to ensure that no conflicting accesses can occur.

Syntax

Definition:

```
HRESULT GetTiffImageRpcUnlocked(
    ULONGLONG& nTiffSize,
    PVOID      pDestBuffer
)
```

Parameters

Name	Type	Description
nTiffSize	ULONGLONG&	Maximum buffer size (in) and actual buffer size (out).
pDestBuffer	PVOID	Pointer to the destination buffer.

Return value

HRESULT [▶ 150]

6.2.2.2.3 Import

Diese Gruppe enthält Interfaces zur Handhabung des Daten-Imports im Zusammenhang mit Bildern.

6.2.2.2.3.1 ITcVnTiffImport

Interface for importing a tiff image.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnTiffImport

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
ImportTiffImage [▶ 1921]	ITcVnTiffImport	Import a tiff image.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.2.3.1.1 ImportTiffImage

Import a tiff image.

Syntax

Definition:

```
HRESULT ImportTiffImage(
    PVOID pTiffImage
)
```

Parameters

Name	Type	Description
pTiffImage	PVOID	Pointer to the tiff file.

Return value

HRESULT [▶ 150]

6.2.2.2.4 ITcIoGevImage

Interface for GigE Vision images.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnImageBase [▶ 1927]
    ITcIoGevImage

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
GetImageData [▶ 1928]	ITcVnImageBase	Gets the data interface (Do not explicitly free the pointer in there!).
GetImageSize [▶ 1928]	ITcVnImageBase	Gets the image size.
GetWidth [▶ 1929]	ITcVnImageBase	Gets the image width (alternatively use F_VN_GetImageWidth).
GetHeight [▶ 1929]	ITcVnImageBase	Gets the image height (alternatively use F_VN_GetImageHeight).
GetXPadding [▶ 1929]	ITcVnImageBase	Gets the horizontal padding.
GetYPadding [▶ 1930]	ITcVnImageBase	Gets the vertical padding.
GetPixelFormat [▶ 1930]	ITcVnImageBase	Gets the pixel format (alternatively use F_VN_GetPixelFormat).
GetImageInfo [▶ 1931]	ITcVnImageBase	Gets a struct containing all common meta infos of the image. This basically encompasses all meta information accessible via this interface. (Alternatively use F_VN_GetImageInfo.)
CheckIfCopy [▶ 1923]	ITcIoGevImage	Check if image data is a copy.
DecreaseHeight [▶ 1923]	ITcIoGevImage	Sets image height to newHeight if it is smaller than the current value.
GetBlockId [▶ 1923]	ITcIoGevImage	Gets the GVSP block ID.
GetGevStatus [▶ 1924]	ITcIoGevImage	Gets the block status.
GetGvspImageInfo [▶ 1924]	ITcIoGevImage	Gets a pointer to the GVSP leader payload.
SetGevStatus [▶ 1924]	ITcIoGevImage	Sets the block status.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.2.4.1 CheckIfCopy

Check if image data is a copy.

Syntax

Definition:

```
HRESULT CheckIfCopy()
```

Return value

HRESULT [▶ 150]

6.2.2.2.4.2 DecreaseHeight

Sets image height to newHeight if it is smaller than the current value.

Syntax

Definition:

```
HRESULT DecreaseHeight(
    ULONG nNewHeight
)
```

Parameters

Name	Type	Description
nNewHeight	ULONG	Height value to set.

Return value

HRESULT [▶ 150]

6.2.2.2.4.3 GetBlockId

Gets the GVSP block ID.

Syntax

Definition:

```
HRESULT GetBlockId(
    ULONGLONG& nBlockId
)
```

Parameters

Name	Type	Description
nBlockId	ULONGLONG&	Returns the block ID.

Return value

HRESULT [▶ 150]

6.2.2.2.4.4 GetGevStatus

Gets the block status.

Syntax

Definition:

```
HRESULT GetGevStatus(
    USHORT& nGevStatus
)
```

Parameters

Name	Type	Description
nGevStatus	USHORT&	Returns the GigE Vision status code.

Return value

HRESULT [▶ 150]

6.2.2.2.4.5 GetGvspImageInfo

Gets a pointer to the GVSP leader payload.

Syntax

Definition:

```
HRESULT GetGvspImageInfo(
    GVSP_IMAGE_INFO** ppGvspImageInfo
)
```

Parameters

Name	Type	Description
ppGvspImageInfo	GVSP_IMAGE_INFO [▶ 1763]**	Pointer to the GVSP meta information.

Return value

HRESULT [▶ 150]

6.2.2.2.4.6 SetGevStatus

Sets the block status.

Syntax

Definition:

```
HRESULT SetGevStatus(
    USHORT nGevStatus
)
```

Parameters

Name	Type	Description
nGevStatus	USHORT	GigE Vision status code.

Return value

HRESULT [▶ 150]

6.2.2.2.5 ITcVnDisplayableImage

Interface for displayable images.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnDisplayableImage

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.2.6 ITcVnImage

Basic interface for images.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnImageBase [▶ 1927]
    ITcVnImage

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
GetImageData [▶ 1928]	ITcVnImageBase	Gets the data interface (Do not explicitly free the pointer in there!).
GetImageSize [▶ 1928]	ITcVnImageBase	Gets the image size.
GetWidth [▶ 1929]	ITcVnImageBase	Gets the image width (alternatively use F_VN_GetImageWidth).
GetHeight [▶ 1929]	ITcVnImageBase	Gets the image height (alternatively use F_VN_GetImageHeight).
GetXPadding [▶ 1929]	ITcVnImageBase	Gets the horizontal padding.
GetYPadding [▶ 1930]	ITcVnImageBase	Gets the vertical padding.
GetPixelFormat [▶ 1930]	ITcVnImageBase	Gets the pixel format (alternatively use F_VN_GetPixelFormat).
GetImageInfo [▶ 1931]	ITcVnImageBase	Gets a struct containing all common meta infos of the image. This basically encompasses all meta information accessible via this interface. (Alternatively use F_VN_GetImageInfo.)
Init [▶ 1926]	ITcVnImage	Initialize an image with an ITcVnImageBase interface.
GetRowPointer [▶ 1927]	ITcVnImage	Gets a pointer to a specific row of an image.
ReleaseRowPointer [▶ 1927]	ITcVnImage	Release the pointer to a specific row of an image.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.2.6.1 Init

Initialize an image with an ITcVnImageBase interface.

Syntax

Definition:

```
HRESULT Init(
    ITcVnImageBase* ipImageBase
)
```

Parameters

Name	Type	Description
ipImageBase	ITcVnImageBase* [▶ 1927]	Image from which to obtain the data.

Return value

HRESULT [▶ 150]

6.2.2.2.6.2 GetRowPointer

Gets a pointer to a specific row of an image.

Syntax

Definition:

```
HRESULT GetRowPointer(
    ULONG  nRowIndex,
    PVOID* ppRow
)
```

Parameters

Name	Type	Description
nRowIndex	ULONG	Row index.
ppRow	PVOID*	Returns a pointer to the requested image row.

Return value

HRESULT [▶ 150]

6.2.2.2.6.3 ReleaseRowPointer

Release the pointer to a specific row of an image.

Syntax

Definition:

```
HRESULT ReleaseRowPointer(
    PVOID* ppRow
)
```

Parameters

Name	Type	Description
ppRow	PVOID*	Pointer to the row pointer to be released.

Return value

HRESULT [▶ 150]

6.2.2.2.7 ITcVnImageBase

Base interface for all image types.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnImageBase

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
GetImageData [▶ 1928]	ITcVnImageBase	Gets the data interface (Do not explicitly free the pointer in there!).
GetImageSize [▶ 1928]	ITcVnImageBase	Gets the image size.
GetWidth [▶ 1929]	ITcVnImageBase	Gets the image width (alternatively use F_VN_GetImageWidth).
GetHeight [▶ 1929]	ITcVnImageBase	Gets the image height (alternatively use F_VN_GetImageHeight).
GetXPadding [▶ 1929]	ITcVnImageBase	Gets the horizontal padding.
GetYPadding [▶ 1930]	ITcVnImageBase	Gets the vertical padding.
GetPixelFormat [▶ 1930]	ITcVnImageBase	Gets the pixel format (alternatively use F_VN_GetPixelFormat).
GetImageInfo [▶ 1931]	ITcVnImageBase	Gets a struct containing all common meta infos of the image. This basically encompasses all meta information accessible via this interface. (Alternatively use F_VN_GetImageInfo.)

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.2.7.1 GetImageData

Gets the data interface (Do not explicitly free the pointer in there!).

Syntax

Definition:

```
HRESULT GetImageData(
    ITcVnData** pipImageData
)
```

Parameters

Name	Type	Description
pipImageData	ITcVnData* [▶ 1935]*	Returns a pointer to the image data interface.

Return value

HRESULT [▶ 150]

6.2.2.2.7.2 GetImageSize

Gets the image size.

Syntax

Definition:

```
HRESULT GetImageSize(
    ULONGLONG& nSize
)
```

Parameters

Name	Type	Description
nSize	ULONGLONG&	Returns the image size in bytes.

Return value

HRESULT [▶ 150]

6.2.2.2.7.3 GetWidth

Gets the image width (alternatively use F_VN_GetImageWidth).

Syntax

Definition:

```
HRESULT GetWidth(
    ULONG& nWidth
)
```

Parameters

Name	Type	Description
nWidth	ULONG&	Returns the image width in pixels.

Return value

HRESULT [▶ 150]

6.2.2.2.7.4 GetHeight

Gets the image height (alternatively use F_VN_GetImageHeight).

Syntax

Definition:

```
HRESULT GetHeight(
    ULONG& nHeight
)
```

Parameters

Name	Type	Description
nHeight	ULONG&	Returns the image height in pixels.

Return value

HRESULT [▶ 150]

6.2.2.2.7.5 GetXPadding

Gets the horizontal padding.

Syntax

Definition:

```
HRESULT GetXPadding(
    USHORT& nXPadding
)
```

Parameters

Name	Type	Description
nXPadding	USHORT&	Returns the horizontal padding in bytes.

Return value

HRESULT [▶ 150]

6.2.2.2.7.6 GetYPadding

Gets the vertical padding.

Syntax

Definition:

```
HRESULT GetYPadding(
    USHORT& nYPadding
)
```

Parameters

Name	Type	Description
nYPadding	USHORT&	Returns the vertical padding in bytes.

Return value

HRESULT [▶ 150]

6.2.2.2.7.7 GetPixelFormat

Gets the pixel format (alternatively use F_VN_GetPixelFormat).

Syntax

Definition:

```
HRESULT GetPixelFormat(
    TcVnPixelFormat& stPixelFormat
)
```

Parameters

Name	Type	Description
stPixelFormat	TcVnPixelFormat [▶ 1780]&	Returns a struct describing the pixel format.

Return value

HRESULT [▶ 150]

6.2.2.2.7.8 GetImageInfo

Gets a struct containing all common meta infos of the image. This basically encompasses all meta information accessible via this interface. (Alternatively use F_VN_GetImageInfo.)

Syntax

Definition:

```
HRESULT GetImageInfo(
    TcVnImageInfo& stImageInfo
)
```

Parameters

Name	Type	Description
stImageInfo	TcVnImageInfo [▶ 1768]&	Returns a struct describing the image.

Return value

HRESULT [▶ 150]

6.2.2.3 Machine Learning

Diese Gruppe enthält Interfaces für Maschinelles Lernen.

6.2.2.3.1 ITcVnColorModel

Interface to train and classify an image color.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnMlModel [▶ 1933]
    ITcVnColorModel

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
GetTypeGuid [▶ 1933]	ITcVnMlModel	Gets the GUID of the model.
GetTypeName [▶ 1933]	ITcVnMlModel	Gets the model type name as a string.
TrainImageColor [▶ 1932]	ITcVnColorModel	Train the color of the provided image.
ClassifyImageColor [▶ 1932]	ITcVnColorModel	Compute the similarity of each image pixel to the trained reference color.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.3.1.1 TrainImageColor

Train the color of the provided image.

Syntax

Definition:

```
HRESULT TrainImageColor(
    ITcVnImage* ipSrcImage,
    ULONG       nDifferentColors,
    ITcVnImage* ipMask,
    ULONG       nSkipPixels
)
```

Parameters

Name	Type	Description
ipSrcImage	ITcVnImage* [▶ 1925]	Source image.
nDifferentColors	ULONG	Number of different colors to differentiate.
ipMask	ITcVnImage* [▶ 1925]	Optional image mask.
nSkipPixels	ULONG	Number of pixels to skip between each evaluated color sample (to achieve a better performance). 0 takes every pixel into account and tends to be more accurate.

Return value

HRESULT [▶ 150]

6.2.2.3.1.2 ClassifyImageColor

Compute the similarity of each image pixel to the trained reference color.

Syntax

Definition:

```
HRESULT ClassifyImageColor(
    ITcVnImage*  ipSrcImage,
    ITcVnImage** pipDestImage,
    float        fVariance,
    float        fLuminanceWeight
)
```

Parameters

Name	Type	Description
ipSrcImage	ITcVnImage* [▶ 1925]	Source image.
pipDestImage	ITcVnImage* [▶ 1925]*	Returns the color similarity.
fVariance	float	Allowed color variance.
fLuminanceWeight	float	Weight the impact of the luminance.

Return value

HRESULT [▶ 150]

6.2.2.3.2 ITcVnMlModel

Interface for a machine learning model.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnMlModel

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
GetTypeGuid [▶ 1933]	ITcVnMlModel	Gets the GUID of the model.
GetTypeName [▶ 1933]	ITcVnMlModel	Gets the model type name as a string.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.3.2.1 GetTypeGuid

Gets the GUID of the model.

Syntax

Definition:

```
HRESULT GetTypeGuid(
    GUID& nTypeGuid
)
```

Parameters

Name	Type	Description
nTypeGuid	GUID&	Returns the GUID of the model.

Return value

HRESULT [▶ 150]

6.2.2.3.2.2 GetTypeName

Gets the model type name as a string.

Syntax

Definition:

```
HRESULT GetTypeName(
    PCHAR  sTypeName,
    USHORT nMaxLen
)
```

Parameters

Name	Type	Description
sTypeName	PCHAR	Returns the model type name as a string.
nMaxLen	USHORT	Maximum string length allowed to be written in sTypeName.

Return value

HRESULT [▶ 150]

6.2.2.4 Miscellaneous

Diese Gruppe enthält sonstige Interfaces.

6.2.2.4.1 ITcSerializableNotification

Interface required for sending serialized objects as ADS notifications.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcSerializableNotification

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
GetComparisonValue [▶ 1934]	ITcSerializableNotification	Gets a value that reflects changes of the underlying object (e.g., the timestamp of the latest object change).
SerializeRpcUnlocked [▶ 1935]	ITcSerializableNotification	Serialize an object into a given buffer.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.4.1.1 GetComparisonValue

Gets a value that reflects changes of the underlying object (e.g., the timestamp of the latest object change).

Syntax

Definition:

```
HRESULT GetComparisonValue(
    LONGLONG& nComparisonValue
)
```

Parameters

Name	Type	Description
nComparisonValue	LONGLONG&	Returns the comparison value.

Return value

HRESULT [▶ 150]

6.2.2.4.1.2 SerializeRpcUnlocked

Serialize an object into a given buffer.

Syntax

Definition:

```
HRESULT SerializeRpcUnlocked(
    ULONGLONG& nBufferSize,
    PVOID      pDestBuffer
)
```

Parameters

Name	Type	Description
nBufferSize	ULONGLONG&	Maximum buffer size (in) and actually used buffer size (out).
pDestBuffer	PVOID	Pointer to the destination buffer.

Return value

HRESULT [▶ 150]

6.2.2.4.2 ITcVnData

Interface for accessing data.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnData

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
CheckIfCopy [▶ 1936]	ITcVnData	Check if the data are a copy (returns S_OK for copies and S_FALSE otherwise).
CheckIfImage [▶ 1936]	ITcVnData	Check if the data belongs to an image (returns S_OK for images and S_FALSE otherwise).
GetSize [▶ 1936]	ITcVnData	Gets the size of the data.
CheckDataPointer [▶ 1937]	ITcVnData	Check if the data pointer is different from 0 (returns S_OK for non-zero data pointers and S_FALSE otherwise).
GetDataPointer [▶ 1937]	ITcVnData	Gets the data pointer (the obtained data pointer must be released by calling ReleaseDataPointer).
ReleaseDataPointer [▶ 1937]	ITcVnData	Release the data pointer.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.4.2.1 CheckIfCopy

Check if the data are a copy (returns S_OK for copies and S_FALSE otherwise).

Syntax

Definition:

```
HRESULT CheckIfCopy()
```

Return value

HRESULT [▶ 150]

6.2.2.4.2.2 CheckIfImage

Check if the data belongs to an image (returns S_OK for images and S_FALSE otherwise).

Syntax

Definition:

```
HRESULT CheckIfImage()
```

Return value

HRESULT [▶ 150]

6.2.2.4.2.3 GetSize

Gets the size of the data.

Syntax

Definition:

```
HRESULT GetSize(
    ULONGLONG& nSize
)
```

Parameters

Name	Type	Description
nSize	ULONGLONG&	Size of the stored data in bytes.

Return value

HRESULT [▶ 150]

6.2.2.4.2.4 CheckDataPointer

Check if the data pointer is different from 0 (returns S_OK for non-zero data pointers and S_FALSE otherwise).

Syntax

Definition:

```
HRESULT CheckDataPointer()
```

Return value

HRESULT [▶ 150]

6.2.2.4.2.5 GetDataPointer

Gets the data pointer (the obtained data pointer must be released by calling ReleaseDataPointer).

Syntax

Definition:

```
HRESULT GetDataPointer(
    PVOID* ppData
)
```

Parameters

Name	Type	Description
ppData	PVOID*	Returns the data pointer.

Return value

HRESULT [▶ 150]

6.2.2.4.2.6 ReleaseDataPointer

Release the data pointer.

Syntax

Definition:

```
HRESULT ReleaseDataPointer(
    PVOID* ppData
)
```

Parameters

Name	Type	Description
ppData	PVOID*	Pointer to the data pointer to be released.

Return value

HRESULT [▶ 150]

6.2.2.4.3 ITcVnTimestamp

Offers an interface for timestamps.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnTimestamp

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
UpdateTimestamp [▶ 1938]	ITcVnTimestamp	Updates the timestamp to the current time
GetTimestamp [▶ 1938]	ITcVnTimestamp	Gets the timestamp

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.4.3.1 UpdateTimestamp

Updates the timestamp to the current time

Syntax

Definition:

```
HRESULT UpdateTimestamp()
```

Return value

HRESULT [▶ 150]

6.2.2.4.3.2 GetTimestamp

Gets the timestamp

Syntax

Definition:

```
HRESULT GetTimestamp(
    LONGLONG& nTimestamp
)
```

Parameters

Name	Type	Description
nTimestamp	LONGLONG&	Returns the timestamp

Return value

HRESULT [▶ 150]

6.2.2.4.4 Schnittstelle ITcUnknown

ITcUnknown definiert die Referenzzählung, sowie das Abfragen einer Referenz auf eine spezifischere Schnittstelle.

Syntax

```
TCOM_DECL_INTERFACE("00000001-0000-0000-e000-000000000064", ITcUnknown)
```

Deklariert in: TcInterfaces.h

Benötigtes include: -

Methoden

Name	Beschreibung
TcAddRef [▶ 1939]	Inkrementiert den Referenzzähler.
TcQueryInterface [▶ 1939]	Abfrage der Referenz an eine implementierte Schnittstelle über der IID.
TcRelease [▶ 1941]	Dekrementiert den Referenzzähler.

Jede TcCOM Schnittstelle ist direkt oder indirekt von ITcUnknown abgeleitet. Demzufolge implementiert jede TcCOM Modulklasse ITcUnknown, weil sie von ITComObject abgeleitet ist.

Die standardmäßige Implementierung von ITcUnknown sorgt dafür, dass das Objekt nach Freigabe der letzten Referenz gelöscht wird. Aus diesem Grunde darf ein Schnittstellenzeiger nach dem Aufruf von TcRelease() nicht dereferenziert werden.

6.2.2.4.4.1 Methode ITcUnknown:TcAddRef

Inkrementiert den Referenzzähler und gibt den neuen Wert zurück.

Syntax

```
ULONG TcAddRef( )
```

Rückgabewert

Daraus resultierender Referenzzählwert.

6.2.2.4.4.2 Methode ITcUnknown:TcQueryInterface

Abfrage eines Schnittstellenzeigers in Bezug auf eine Schnittstelle, die per Interface ID (IID) gegeben ist.

Syntax

```
HRESULT TcQueryInterface(RITCID iid, PPVOID pipItf )
```

Rückgabewert

Bei Erfolg wird S_OK („0“) oder ein anderer positiver Wert zurückgegeben, vgl. Rückgabewerte. Erweiterte Meldungen beziehen sich dabei insbesondere auf die Spalte HRESULT in ADS Return Codes [▶ 2966].

Wenn die verlangte Schnittstelle nicht verfügbar ist, gibt die Methode ADSERR_DEVICE_NOINTERFACE zurück.

Parameter

Name	Typ	Beschreibung
iid	RITCID	Schnittstelle IID.
pipItf	PPVOID	Zeiger auf Schnittstellenzeiger. Wird gesetzt, wenn der verlangte Schnittstellentyp von der entsprechenden Instanz verfügbar ist.

Beschreibung

Abfrage der Referenz an eine implementierte Schnittstelle über der IID. Es wird empfohlen, Smart Pointer zu verwenden, um Schnittstellenzeiger zu initialisieren und zu halten.

Variante 1:

```
HRESULT GetTraceLevel(ITcUnkown* ip, TcTraceLevel& tl)
{
HRESULT hr = S_OK;
if (ip != NULL)
{
ITComObjectPtr spObj;
hr = ip->TcQueryInterface(spObj.GetIID(), &spObj);
if (SUCCEEDED(hr))
{
hr = spObj->TcGetObjPara(PID_TcTraceLevel, &tl, sizeof(tl));
}
return hr;
}
}
```

Die mit dem Smart Pointer verbundene Schnittstellen-ID kann in TcQueryInterface als Parameter verwendet werden. Der Operator „&“ wird den Zeiger auf die interne Schnittstellen-Zeiger-Membervariable des Smart Pointers zurückgeben. Variante 1 geht davon aus, dass der Schnittstellenzeiger initialisiert ist, wenn TcQueryInterface Erfolg anzeigt. Wenn der Bereich bleibt, dann gibt der Destructor des Smart Pointers spObj die Referenz frei.

Variante 2:

```
HRESULT GetTraceLevel(ITcUnkown* ip, TcTraceLevel& tl)
{
HRESULT hr = S_OK;
ITComObjectPtr spObj = ip;
if (spObj != NULL)
{
spObj->TcGetObjParam(PID_TcTraceLevel, &tl);
}
else
{
hr = ADS_E_NOINTERFACE;
}
return hr;
}
```

Wenn der Schnittstellenzeiger ip dem Smart Pointer spObj zugewiesen wird, dann wird die TcQueryInterface-Methode implizit aufgerufen mit IID_ITComObject auf der Instanz, auf die ip verweist. Dies führt zu einem kürzeren Code, aber der ursprüngliche Return-Code von TcQueryInterface geht verloren.

6.2.2.4.4.3 Methode ITcUnknown:TcRelease

Diese Methode dekrementiert den Referenzzähler.

Syntax

```
ULONG TcRelease( )
```

Rückgabewert

Daraus resultierender Referenzzählwert.

Beschreibung

Dekrementiert den Referenzzähler und gibt den neuen Wert zurück.

Wenn der Referenzzähler 0 wird, löscht das Objekt sich selbst.

6.2.2.5 Neural Network

6.2.2.5.1 ITcVnNeuralNetwork

Interface for neural networks.

Inheritance Hierarchy

ITcUnknown [▶ 1939]
  ITcVnMlModel [▶ 1933]
    ITcVnNeuralNetwork

Methods

Name	Origin	Description
TcAddRef [▶ 1939]	ITcUnknown	Increments the reference counter.
TcQueryInterface [▶ 1939]	ITcUnknown	Get a reference to an implemented interface.
TcRelease [▶ 1941]	ITcUnknown	Decrements the reference counter.
GetTypeGuid [▶ 1933]	ITcVnMlModel	Gets the GUID of the model.
GetTypeName [▶ 1933]	ITcVnMlModel	Gets the model type name as a string.
Execute [▶ 1942]	ITcVnNeuralNetwork	Execute a neural network using the provided image as input and provide the output of one output layer. The method doesn't perform any pre-processing or transformation.
Execute_SingleLayerOutput [▶ 1942]	ITcVnNeuralNetwork	Execute a neural network using the provided image as input and provide the output of the specified output layer. The method doesn't perform any pre-processing or transformation.
Execute_MultiLayerOutput [▶ 1943]	ITcVnNeuralNetwork	Execute a neural network using the provided image as input and provide the output of the specified output layers. If sLayerNames is empty, the method returns the output of the unconnected output layers. The method doesn't perform any pre-processing or transformation.
GetLayerNames [▶ 1944]	ITcVnNeuralNetwork	Get the name of all layers in the model. If the bOnlyOuput flag is set to true, only the unconnected output layer names are returned.

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.2.5.1.1 Execute

Execute a neural network using the provided image as input and provide the output of one output layer. The method doesn't perform any pre-processing or transformation.

Syntax

Definition:

```
HRESULT Execute(
    ITcVnImage*  ipSrcImage,
    ITcVnImage** pipDestImage
)
```

Parameters

Name	Type	Description
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (The image layout must be adapted to match the model's expected input format using e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage functions if necessary.)
pipDestImage	ITcVnImage* [▶ 1925]*	Destination image (An appropriate destination image based on the output layer is created. To process the destination image, use e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage if necessary.

Return value

HRESULT [▶ 150]

6.2.2.5.1.2 Execute_SingleLayerOutput

Execute a neural network using the provided image as input and provide the output of the specified output layer. The method doesn't perform any pre-processing or transformation.

Syntax

Definition:

```
HRESULT Execute_SingleLayerOutput(
    ITcVnImage*  ipSrcImage,
    ITcVnImage** pipDestImage,
    PCCH         sLayerName
)
```

Parameters

Name	Type	Description
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (The image layout must be adapted to match the model's expected input format using e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage functions if necessary.)
pipDestImage	ITcVnImage* [▶ 1925]*	Destination image (An appropriate destination image based on the output layer is created. To process the destination image, use e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage if necessary.
sLayerName	PCCH	Specify the name of the layer for which the output is required. If empty, the output of one output layer is provided.

Return value

HRESULT [▶ 150]

6.2.2.5.1.3 Execute_MultiLayerOutput

Execute a neural network using the provided image as input and provide the output of the specified output layers. If sLayerNames is empty, the method returns the output of the unconnected output layers. The method doesn't perform any pre-processing or transformation.

Syntax

Definition:

```
HRESULT Execute_MultiLayerOutput(
    ITcVnImage*      ipSrcImage,
    ITcVnContainer** pipDestImages,
    ITcVnContainer*  ipLayerNames
)
```

Parameters

Name	Type	Description
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (The image layout must be adapted to match the model's expected input format using e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage functions if necessary.)
pipDestImages	ITcVnContainer* [▶ 1890]*	Container of images containing the output from the specified layers
ipLayerNames	ITcVnContainer* [▶ 1890]	Container of layer names whose outputs to be returned (ContainerType_Vector_String_SINT). If set to zero, the container ipDestImages holds the outputs of all the unconnected output layers.

Return value

HRESULT [▶ 150]

6.2.2.5.1.4 GetLayerNames

Get the name of all layers in the model. If the bOnlyOuput flag is set to true, only the unconnected output layer names are returned.

Syntax

Definition:

```
HRESULT GetLayerNames(
    ITcVnContainer** pipLayerNames,
    bool             bOnlyOutputs
)
```

Parameters

Name	Type	Description
pipLayerNames	ITcVnContainer* [▶ 1890]*	Container with the layers names (ContainerType_Vector_String_SINT)
bOnlyOutputs	bool	If true, only the unconnected output layer names are returned

Return value

HRESULT [▶ 150]

6.2.3 Functions

Die Bildverarbeitungsalgorithmen sind in Funktionen gekapselt. Komplexere Algorithmen wurden als Funktionsbaustein umgesetzt und sind unter ImageProcessing [▶ 2779] zu finden.

Aufbau eines Funktionsaufrufs

Funktionen sind wie folgt aufgebaut:

```
hr := ProcessImage (hr, <...>, <...>);

Status variable of type HRESULT
Parameters and return values as references
Optional parameters with default values
```

hr	Statusvariable [▶ 150] vom Typ HRESULT. Hiermit kann der Ausführungsstatus jeder Funktion nachvollzogen werden.
Parameter <...>	Funktionsparameter erfüllen im Wesentlichen folgenden Zweck: • Eingangsparameter (z. B. Bilder oder Container), die verarbeitet werden sollen. • Parameter zur Einstellung des Algorithmus • Parameter als Referenz zur Rückgabe der Ergebnisse
Optional parameter <...>	Diese Parameter werden mit Default-Werten initialisiert, daher ist die Angabe optional.

Gruppierung von Funktionen

Alle Funktionen der TwinCAT Vision Bibliothek sind thematisch in die folgenden Gruppen sortiert:

Bilder

- Basic Image Operations [▶ 2224]
- Algebraic Image Operations [▶ 2005]
- Image Segmentation [▶ 2567]
- Image Color and Contrast Processing [▶ 2528]
- Image Analysis [▶ 2497]
- Image Filtering [▶ 2549]

Container

- Basic Container Operations [▶ 2041]
- Algebraic Container Operations [▶ 1971]
- Container Statistics [▶ 2276]

Konturen

- Contour Analysis [▶ 2348]

Vermessung

- Measurement [▶ 2650]
- Geometric and Coordinate Transformations [▶ 2423]

Code-Qualität

- Code Quality [▶ 2258]

Code-Lesung

- Code Reading [▶ 2265]

Optische Zeichenerkennung (OCR)

- Optical Character Recognition [▶ 2726]

Zeichnen und Text

- Drawing [▶ 2379]

Machine Learning

- MachineLearning [▶ 2609]
- Neural Network [▶ 2708]

Fortgeschrittene Funktionen

- Fourier Analysis [▶ 2412]
- Keypoint Features [▶ 2574]

Sonstiges

- Control Functions [▶ 2376]
- Miscellaneous [▶ 2683]

Expert-Funktionen

Zu vielen Funktionen existieren Expert-Varianten. Diese enthalten zusätzliche Parameter und sind in der SPS-Bibliothek durch das Namenssuffix „`Exp`“ gekennzeichnet. In C++ sind diese zusätzlichen Parameter mit Default-Werten belegt und können beim Aufruf, wenn nicht benötigt, weggelassen werden.

Laufzeitverhalten

Die Ausführungsdauer einiger Bildverarbeitungsalgorithmen hängt neben ihrer Parametrierung auch vom Bildinhalt ab. Sie ist daher nicht deterministisch. Um dennoch zyklisches Echtzeitverhalten zu gewährleisten, können Watchdogs [▶ 156] auf diese Funktionen oder Funktionsgruppen angewandt werden, die die Funktionen nach einer vorgegebenen Zeit abbrechen.

⚠ WARNUNG

TwinCAT-Absturz bei falscher Verwendung von Funktionen

Bei falscher Verwendung von einigen Funktionen kann es zu Fehlern (und daraus resultierenden TwinCAT-Abstürzen) kommen. Diese werden nicht innerhalb der Funktion abgefangen, da entsprechende Überprüfungen die Ausführungszeit wesentlich verlängern würden. Sie müssen also selbst darauf achten, die Funktionen ordnungsgemäß zu verwenden! Z. B. empfehlen wir, eine Validierung der Eingabeparameter durchzuführen.

6.2.3.1 License Overview

Folgend ist die Zugehörigkeit der TwinCAT Vision Funktionen zu verfügbaren Lizenzen (siehe Lizenzmodell [▶ 12]) aufgelistet. Daraus ist ersichtlich, welche Lizenz für die Anwendung bestimmter Funktionen benötigt wird.

TF7100

Die Basis-Lizenz TC3 Vision Base (TF7100) wird immer benötigt.

TC3 Vision Base

AdaptiveThreshold [▶ 2568]

AddContainers [▶ 1995]

AddImages [▶ 2007]

AddImagesWeighted [▶ 2007]

AddScalarToImage [▶ 2008]

AddToContainerElements (LONG) [▶ 1972]

AddToContainerElements (SHORT) [▶ 1973]

AddToContainerElements (double) [▶ 1973]

AddToContainerElements (float) [▶ 1974]

AddToContainerElements (char) [▶ 1975]

AddToContainerElements (TcVnKeyPoint) [▶ 1975]

AddToContainerElements (TcVnPoint2_DINT) [▶ 1976]

AddToContainerElements (TcVnPoint2_LREAL) [▶ 1977]

AddToContainerElements (TcVnPoint2_REAL) [▶ 1977]

AddToContainerElements (TcVnPoint3_LREAL) [▶ 1978]

AddToContainerElements (TcVnPoint3_REAL) [▶ 1979]

AddToContainerElements (TcVnRectangle_DINT) [▶ 1979]

AddToContainerElements (TcVnRotatedRectangle) [▶ 1995]

AddToContainerElements (TcVnVector2_DINT) [▶ 1980]

AddToContainerElements (TcVnVector2_INT) [▶ 1981]

AddToContainerElements (TcVnVector2_REAL) [▶ 1981]

AddToContainerElements (TcVnVector2_SINT) [▶ 1982]

AddToContainerElements (TcVnVector2_UINT) [▶ 1983]

AddToContainerElements (TcVnVector2_USINT) [▶ 1983]

AddToContainerElements (TcVnVector3_INT) [▶ 1984]

AddToContainerElements (TcVnVector3_REAL) [▶ 1985]

AddToContainerElements (TcVnVector3_SINT) [▶ 1985]

AddToContainerElements (TcVnVector3_UINT) [▶ 1986]

AddToContainerElements (TcVnVector3_USINT) [▶ 1987]

AddToContainerElements (TcVnVector4_DINT) [▶ 1987]

AddToContainerElements (TcVnVector4_INT) [▶ 1988]

AddToContainerElements (TcVnVector4_LREAL) [▶ 1989]

AddToContainerElements (TcVnVector4_REAL) [▶ 1994]

AddToContainerElements (TcVnVector4_SINT) [▶ 1989]

AddToContainerElements (TcVnVector4_UINT) [▶ 1990]

AddToContainerElements (TcVnVector4_USINT) [▶ 1991]

AddToContainerElements (ULONG) [▶ 1991]

AddToContainerElements (USHORT) [▶ 1992]

AddToContainerElements (ULONGLONG) [▶ 1993]

AddToContainerElements (unsigned char) [▶ 1993]

AddVectorToImage [▶ 2009]

AdjustActiveContour [▶ 2498]

AdvanceIterator [▶ 2205]

AlignRotatedImageRegion [▶ 2457]

AppendToContainer (LONG) [▶ 2043]

AppendToContainer (SHORT) [▶ 2043]

AppendToContainer (ITcVnContainer*) [▶ 2044]

AppendToContainer (ITcVnForwardIterator*) [▶ 2045]

AppendToContainer (ITcVnImage*) [▶ 2045]

AppendToContainer (double) [▶ 2046]

AppendToContainer (float) [▶ 2047]

AppendToContainer (char) [▶ 2047]

AppendToContainer (TcVnDMatch) [▶ 2048]

AppendToContainer (TcVnKeyPoint) [▶ 2049]

AppendToContainer (TcVnPoint2_DINT) [▶ 2049]

AppendToContainer (TcVnPoint2_LREAL) [▶ 2050]

AppendToContainer (TcVnPoint2_REAL) [▶ 2051]

AppendToContainer (TcVnPoint3_LREAL) [▶ 2051]

AppendToContainer (TcVnPoint3_REAL) [▶ 2052]

AppendToContainer (TcVnRectangle_DINT) [▶ 2053]

AppendToContainer (TcVnRotatedRectangle) [▶ 2068]

AppendToContainer (TcVnVector2_DINT) [▶ 2053]

AppendToContainer (TcVnVector2_INT) [▶ 2054]

AppendToContainer (TcVnVector2_REAL) [▶ 2055]

AppendToContainer (TcVnVector2_SINT) [▶ 2055]

AppendToContainer (TcVnVector2_UINT) [▶ 2056]

AppendToContainer (TcVnVector2_USINT) [▶ 2057]

AppendToContainer (TcVnVector3_INT) [▶ 2057]

AppendToContainer (TcVnVector3_REAL) [▶ 2058]

AppendToContainer (TcVnVector3_SINT) [▶ 2059]

AppendToContainer (TcVnVector3_UINT) [▶ 2059]

AppendToContainer (TcVnVector3_USINT) [▶ 2060]

AppendToContainer (TcVnVector4_DINT) [▶ 2061]

AppendToContainer (TcVnVector4_INT) [▶ 2061]

AppendToContainer (TcVnVector4_LREAL) [▶ 2062]

AppendToContainer (TcVnVector4_REAL) [▶ 2067]

AppendToContainer (TcVnVector4_SINT) [▶ 2063]

AppendToContainer (TcVnVector4_UINT) [▶ 2063]

AppendToContainer (TcVnVector4_USINT) [▶ 2064]

AppendToContainer (ULONG) [▶ 2065]

AppendToContainer (USHORT) [▶ 2065]

AppendToContainer (ULONGLONG) [▶ 2066]

AppendToContainer (unsigned char) [▶ 2067]

ApplyColorMap [▶ 2528]

ApplyLut [▶ 2529]

ApplyRotationToAffineTransformation [▶ 2458]

ApplyScalingToAffineTransformation [▶ 2459]

ApplyTranslationToAffineTransformation [▶ 2460]

ApplyYAxisInversionToAffineTransformation [▶ 2461]

ApproximatePolygon [▶ 2349]

BilateralFilter [▶ 2550]

BitwiseAndContainers [▶ 1996]

BitwiseAndImages [▶ 2010]

BitwiseAndImages (Exp) [▶ 2011]

BitwiseAndScalarWithImage [▶ 2012]

BitwiseAndVectorWithImage [▶ 2012]

BitwiseNotContainer [▶ 1997]

BitwiseNotImage [▶ 2013]

BitwiseNotImage (Exp) [▶ 2014]

BitwiseOrContainers [▶ 1997]

BitwiseOrImages [▶ 2014]

BitwiseOrImages (Exp) [▶ 2015]

BitwiseOrScalarWithImage [▶ 2016]

BitwiseOrVectorWithImage [▶ 2017]

BitwiseXorContainers [▶ 1998]

BitwiseXorImages [▶ 2018]

BitwiseXorImages (Exp) [▶ 2018]

BitwiseXorScalarWithImage [▶ 2019]

BitwiseXorVectorWithImage [▶ 2020]

BlendImages [▶ 2021]

BoxFilter [▶ 2551]

BrightBorderObjects [▶ 2552]

CalibrateCameraPlanar [▶ 2433]

CannyEdgeDetection [▶ 2512]

CheckColorRange [▶ 2570]

CheckFunctionInitialization [▶ 2684]

CheckIfEmpty [▶ 2205]

CheckIfIteratorIsAtEnd [▶ 2206]

CheckIfPointIsInsideContour [▶ 2350]

Clahe [▶ 2530]

ClipLineToBoundary [▶ 2684]

ClipLineToBoundary (ITcVnImage*) [▶ 2685]

ClipLineToBoundary (TcVnRectangle_DINT) [▶ 2686]

CombineImageChannels [▶ 2226]

ConnectedComponents [▶ 2513]

ConnectedComponentsWithStats [▶ 2514]

ContainerAverage [▶ 2340]

ContainerAverageElementwise (2) [▶ 2341]

ContainerAverageElementwise (3) [▶ 2342]

ContainerAverageElementwise (4) [▶ 2342]

ContainerAverageVariance [▶ 2343]

ContainerAverageVarianceElementwise (2) [▶ 2344]

ContainerAverageVarianceElementwise (3) [▶ 2344]

ContainerAverageVarianceElementwise (4) [▶ 2345]

ContourArea [▶ 2351]

ContourCenterOfMass [▶ 2352]

ContourCircularity [▶ 2353]

ContourConvexity [▶ 2354]

ContourEccentricity [▶ 2355]

ContourElongation [▶ 2356]

ContourExtremePoint [▶ 2357]

ContourInertiaRatio [▶ 2357]

ContourMoments [▶ 2358]

ContourOrientation [▶ 2359]

ContourPerimeter [▶ 2360]

ContourRoundness [▶ 2361]

ConvertCartesianToPolarAngleImage [▶ 2462]

ConvertCartesianToPolarAngles [▶ 2463]

ConvertCartesianToPolarImages [▶ 2464]

ConvertCartesianToPolarMagnitudeImage [▶ 2465]

ConvertCartesianToPolarMagnitudes [▶ 2466]

ConvertCartesianToPolarPoints [▶ 2467]

ConvertColorSpace [▶ 2531]

ConvertContainerType [▶ 2207]

ConvertDataLayout [▶ 2709]

ConvertElementType [▶ 2226]

ConvertITcUnknownToITcVnBitmapExport [▶ 2687]

ConvertITcUnknownToITcVnContainer [▶ 2688]

ConvertITcUnknownToITcVnImage [▶ 2688]

ConvertITcUnknownToITcVnMlModel [▶ 2689]

ConvertPolarToCartesianImages [▶ 2468]

ConvertPolarToCartesianPoints [▶ 2469]

ConvexHullPoints [▶ 2362]

ConvexityDefects [▶ 2363]

CopyContainer [▶ 2207]

CopyContainerElementsConditional (ITcVnContainer*) [▶ 2208]

CopyContainerElementsConditional (ITcVnForwardIterator*) [▶ 2209]

CopyImage [▶ 2227]

CopyImageRegion [▶ 2228]

CopyImageRegionToRegion [▶ 2229]

CopyIntoDisplayableImage [▶ 2230]

CountNonZeroPixels [▶ 2516]

CreateAssociatedImage [▶ 2231]

CreateBandpassButterworthFilter [▶ 2412]

CreateBandpassGaussianFilter [▶ 2413]

CreateBandrejectButterworthFilter [▶ 2414]

CreateBandrejectGaussianFilter [▶ 2416]

CreateContainer [▶ 2209]

CreateContainerFromArray [▶ 2210]

CreateEmptyImage [▶ 2231]

CreateHighpassButterworthFilter [▶ 2417]

CreateHighpassGaussianFilter [▶ 2418]

CreateImage [▶ 2232]

CreateImageAndSetPixels [▶ 2233]

CreateImageFromArray [▶ 2234]

CreateLowpassButterworthFilter [▶ 2419]

CreateLowpassGaussianFilter [▶ 2420]

CreateNDimensionalImage [▶ 2253]

CreateNDimensionalImageAndSetPixels [▶ 2254]

CreateNDimensionalImageFromArray [▶ 2255]

CreateStructuringElement [▶ 2552]

CustomElementWiseContainerOperation (ITcVnContainer*) [▶ 2364]

CustomElementWiseContainerOperation (ITcVnForwardIterator*) [▶ 2365]

CustomFilter [▶ 2553]

DarkBorderObjects [▶ 2554]

DecomposeAffineTransformation [▶ 2470]

DecomposeHomography [▶ 2471]

DeinitializeFunction [▶ 2690]

DetectBlobs [▶ 2499]

Dft [▶ 2421]

DistanceTransformation [▶ 2516]

DivideContainers [▶ 1999]

DivideImageByScalar [▶ 2021]

DivideImageByVector [▶ 2022]

DivideImages [▶ 2023]

DivideScalarByImage [▶ 2023]

DivideVectorByImage [▶ 2024]

DoubleThreshold [▶ 2570]

DrawArrow [▶ 2380]

DrawArrow (TcVnVector4_DINT) [▶ 2381]

DrawCircle [▶ 2382]

DrawCircles [▶ 2383]

DrawCircularArc [▶ 2384]

DrawComponents [▶ 2385]

DrawContours [▶ 2386]

DrawEllipse [▶ 2387]

DrawLine [▶ 2389]

DrawLine (TcVnVector4_DINT) [▶ 2390]

DrawLine (TcVnVector4_LREAL) [▶ 2391]

DrawLines [▶ 2392]

DrawOrientation [▶ 2396]

DrawOrientation (Exp) [▶ 2396]

DrawPoint [▶ 2397]

DrawPoints [▶ 2398]

DrawPolygon [▶ 2399]

DrawRectangle [▶ 2400]

DrawRectangle (TcVnRectangle_DINT) [▶ 2401]

DrawRectangle (TcVnRectangle_UDINT) [▶ 2402]

DrawRotatedRectangle [▶ 2403]

ElementwiseExp [▶ 2025]

ElementwiseLog [▶ 2026]

EnclosingCircle [▶ 2366]

EnclosingRectangle [▶ 2367]

EnclosingTriangle [▶ 2367]

EraseFromContainer [▶ 2211]

ExportContainer [▶ 2212]

ExportContainer_String [▶ 2213]

ExportContainerSize [▶ 2213]

ExportImage [▶ 2235]

ExportImageAsBmp [▶ 2236]

ExportImageAsBmpSize [▶ 2237]

ExportImageSize [▶ 2238]

ExportSubContainer [▶ 2214]

ExportSubContainer_String [▶ 2215]

ExportSubContainerSize [▶ 2216]

ExtractContainerRange [▶ 2216]

FillCircle [▶ 2404]

FillContainer (LONG) [▶ 2069]

FillContainer (SHORT) [▶ 2069]

FillContainer (ITcVnImage*) [▶ 2070]

FillContainer (double) [▶ 2071]

FillContainer (float) [▶ 2071]

FillContainer (char) [▶ 2072]

FillContainer (TcVnDMatch) [▶ 2073]

FillContainer (TcVnKeyPoint) [▶ 2073]

FillContainer (TcVnPoint2_DINT) [▶ 2074]

FillContainer (TcVnPoint2_LREAL) [▶ 2075]

FillContainer (TcVnPoint2_REAL) [▶ 2075]

FillContainer (TcVnPoint3_LREAL) [▶ 2076]

FillContainer (TcVnPoint3_REAL) [▶ 2077]

FillContainer (TcVnRectangle_DINT) [▶ 2077]

FillContainer (TcVnRotatedRectangle) [▶ 2093]

FillContainer (TcVnVector2_DINT) [▶ 2078]

FillContainer (TcVnVector2_INT) [▶ 2079]

FillContainer (TcVnVector2_REAL) [▶ 2079]

FillContainer (TcVnVector2_SINT) [▶ 2080]

FillContainer (TcVnVector2_UINT) [▶ 2081]

FillContainer (TcVnVector2_USINT) [▶ 2081]

FillContainer (TcVnVector3_INT) [▶ 2082]

FillContainer (TcVnVector3_REAL) [▶ 2083]

FillContainer (TcVnVector3_SINT) [▶ 2083]

FillContainer (TcVnVector3_UINT) [▶ 2084]

FillContainer (TcVnVector3_USINT) [▶ 2085]

FillContainer (TcVnVector4_DINT) [▶ 2085]

FillContainer (TcVnVector4_INT) [▶ 2086]

FillContainer (TcVnVector4_LREAL) [▶ 2087]

FillContainer (TcVnVector4_REAL) [▶ 2092]

FillContainer (TcVnVector4_SINT) [▶ 2087]

FillContainer (TcVnVector4_UINT) [▶ 2088]

FillContainer (TcVnVector4_USINT) [▶ 2089]

FillContainer (ULONG) [▶ 2089]

FillContainer (USHORT) [▶ 2090]

FillContainer (ULONGLONG) [▶ 2091]

FillContainer (unsigned char) [▶ 2091]

FillContainer (LONG) [▶ 2093]

FillContainer (SHORT) [▶ 2094]

FillContainer (ITcVnImage*) [▶ 2095]

FillContainer (double) [▶ 2095]

FillContainer (float) [▶ 2096]

FillContainer (char) [▶ 2097]

FillContainer (TcVnDMatch) [▶ 2098]

FillContainer (TcVnKeyPoint) [▶ 2098]

FillContainer (TcVnPoint2_DINT) [▶ 2099]

FillContainer (TcVnPoint2_LREAL) [▶ 2100]

FillContainer (TcVnPoint2_REAL) [▶ 2100]

FillContainer (TcVnPoint3_LREAL) [▶ 2101]

FillContainer (TcVnPoint3_REAL) [▶ 2102]

FillContainer (TcVnRectangle_DINT) [▶ 2103]

FillContainer (TcVnRotatedRectangle) [▶ 2119]

FillContainer (TcVnVector2_DINT) [▶ 2103]

FillContainer (TcVnVector2_INT) [▶ 2104]

FillContainer (TcVnVector2_REAL) [▶ 2105]

FillContainer (TcVnVector2_SINT) [▶ 2105]

FillContainer (TcVnVector2_UINT) [▶ 2106]

FillContainer (TcVnVector2_USINT) [▶ 2107]

FillContainer (TcVnVector3_INT) [▶ 2108]

FillContainer (TcVnVector3_REAL) [▶ 2108]

FillContainer (TcVnVector3_SINT) [▶ 2109]

FillContainer (TcVnVector3_UINT) [▶ 2110]

FillContainer (TcVnVector3_USINT) [▶ 2110]

FillContainer (TcVnVector4_DINT) [▶ 2111]

FillContainer (TcVnVector4_INT) [▶ 2112]

FillContainer (TcVnVector4_LREAL) [▶ 2113]

FillContainer (TcVnVector4_REAL) [▶ 2118]

FillContainer (TcVnVector4_SINT) [▶ 2113]

FillContainer (TcVnVector4_UINT) [▶ 2114]

FillContainer (TcVnVector4_USINT) [▶ 2115]

FillContainer (ULONG) [▶ 2115]

FillContainer (USHORT) [▶ 2116]

FillContainer (ULONGLONG) [▶ 2117]

FillContainer (unsigned char) [▶ 2118]

FillContours [▶ 2404]

FillEllipse [▶ 2405]

FillHoles [▶ 2555]

FillPolygon [▶ 2406]

FillRectangle [▶ 2407]

FillRotatedRectangle [▶ 2408]

FindContourHierarchy (Exp) [▶ 2500]

FindContours [▶ 2501]

FitEllipse [▶ 2368]

FitLine [▶ 2369]

FlipImage [▶ 2475]

FourierDescriptors [▶ 2371]

FuseImages [▶ 2239]

FuseImagesArray [▶ 2240]

GaussianFilter [▶ 2556]

GenerateAffineTransformationUnitMatrix2D [▶ 2476]

GenerateColorMap [▶ 2532]

GenerateCustomColorMap [▶ 2533]

GetAffineTransformation [▶ 2476]

GetAffineTransformation2D [▶ 2477]

GetAt (LONG) [▶ 2120]

GetAt (SHORT) [▶ 2121]

GetAt (ITcVnContainer*) [▶ 2121]

GetAt (ITcVnImage*) [▶ 2122]

GetAt (double) [▶ 2123]

GetAt (float) [▶ 2123]

GetAt (char) [▶ 2124]

GetAt (TcVnDMatch) [▶ 2125]

GetAt (TcVnKeyPoint) [▶ 2125]

GetAt (TcVnPoint2_DINT) [▶ 2126]

GetAt (TcVnPoint2_LREAL) [▶ 2127]

GetAt (TcVnPoint2_REAL) [▶ 2127]

GetAt (TcVnPoint3_LREAL) [▶ 2128]

GetAt (TcVnPoint3_REAL) [▶ 2129]

GetAt (TcVnRectangle_DINT) [▶ 2130]

GetAt (TcVnRotatedRectangle) [▶ 2146]

GetAt (TcVnVector2_DINT) [▶ 2130]

GetAt (TcVnVector2_INT) [▶ 2131]

GetAt (TcVnVector2_REAL) [▶ 2132]

GetAt (TcVnVector2_SINT) [▶ 2132]

GetAt (TcVnVector2_UINT) [▶ 2133]

GetAt (TcVnVector2_USINT) [▶ 2134]

GetAt (TcVnVector3_INT) [▶ 2135]

GetAt (TcVnVector3_REAL) [▶ 2135]

GetAt (TcVnVector3_SINT) [▶ 2136]

GetAt (TcVnVector3_UINT) [▶ 2137]

GetAt (TcVnVector3_USINT) [▶ 2137]

GetAt (TcVnVector4_DINT) [▶ 2138]

GetAt (TcVnVector4_INT) [▶ 2139]

GetAt (TcVnVector4_LREAL) [▶ 2140]

GetAt (TcVnVector4_REAL) [▶ 2145]

GetAt (TcVnVector4_SINT) [▶ 2140]

GetAt (TcVnVector4_UINT) [▶ 2141]

GetAt (TcVnVector4_USINT) [▶ 2142]

GetAt (ULONG) [▶ 2142]

GetAt (USHORT) [▶ 2143]

GetAt (ULONGLONG) [▶ 2144]

GetAt (unsigned char) [▶ 2145]

GetConnectedComponent [▶ 2517]

GetContainer [▶ 2217]

GetContainer (Exp) [▶ 2218]

GetForwardIterator [▶ 2218]

GetImageChannel [▶ 2240]

GetImageHeight [▶ 2241]

GetImageInfo [▶ 2242]

GetImageWidth [▶ 2242]

GetLayoutInfo [▶ 2255]

GetNumberOfElements [▶ 2219]

GetPerspectiveTransformation [▶ 2479]

GetPixel [▶ 2243]

GetPixelFormat [▶ 2244]

GetRandomAccessIterator [▶ 2220]

GetRoi [▶ 2244]

GetTimestamp [▶ 2690]

Histogram [▶ 2534]

HistogramEqualization [▶ 2535]

Homography [▶ 2479]

HoughCircles [▶ 2503]

HoughLines [▶ 2505]

HoughLinesP [▶ 2506]

HuMomentInvariants [▶ 2691]

ImageAbsoluteAverage [▶ 2526]

ImageAverage [▶ 2518]

ImageAverageStdDev [▶ 2519]

ImageCenterOfMass [▶ 2520]

ImageMedian [▶ 2521]

ImageMoments [▶ 2522]

IncrementIterator [▶ 2220]

InitMatrixStruct [▶ 2692]

InsertIntoContainer (LONG) [▶ 2147]

InsertIntoContainer (SHORT) [▶ 2148]

InsertIntoContainer (ITcVnContainer*) [▶ 2148]

InsertIntoContainer (ITcVnForwardIterator*) [▶ 2149]

InsertIntoContainer (ITcVnImage*) [▶ 2150]

InsertIntoContainer (double) [▶ 2150]

InsertIntoContainer (float) [▶ 2151]

InsertIntoContainer (char) [▶ 2152]

InsertIntoContainer (TcVnDMatch) [▶ 2153]

InsertIntoContainer (TcVnKeyPoint) [▶ 2153]

InsertIntoContainer (TcVnPoint2_DINT) [▶ 2154]

InsertIntoContainer (TcVnPoint2_LREAL) [▶ 2155]

InsertIntoContainer (TcVnPoint2_REAL) [▶ 2156]

InsertIntoContainer (TcVnPoint3_LREAL) [▶ 2156]

InsertIntoContainer (TcVnPoint3_REAL) [▶ 2157]

InsertIntoContainer (TcVnRectangle_DINT) [▶ 2158]

InsertIntoContainer (TcVnRotatedRectangle) [▶ 2175]

InsertIntoContainer (TcVnVector2_DINT) [▶ 2159]

InsertIntoContainer (TcVnVector2_INT) [▶ 2159]

InsertIntoContainer (TcVnVector2_REAL) [▶ 2160]

InsertIntoContainer (TcVnVector2_SINT) [▶ 2161]

InsertIntoContainer (TcVnVector2_UINT) [▶ 2162]

InsertIntoContainer (TcVnVector2_USINT) [▶ 2162]

InsertIntoContainer (TcVnVector3_INT) [▶ 2163]

InsertIntoContainer (TcVnVector3_REAL) [▶ 2164]

InsertIntoContainer (TcVnVector3_SINT) [▶ 2165]

InsertIntoContainer (TcVnVector3_UINT) [▶ 2165]

InsertIntoContainer (TcVnVector3_USINT) [▶ 2166]

InsertIntoContainer (TcVnVector4_DINT) [▶ 2167]

InsertIntoContainer (TcVnVector4_INT) [▶ 2168]

InsertIntoContainer (TcVnVector4_LREAL) [▶ 2168]

InsertIntoContainer (TcVnVector4_REAL) [▶ 2174]

InsertIntoContainer (TcVnVector4_SINT) [▶ 2169]

InsertIntoContainer (TcVnVector4_UINT) [▶ 2170]

InsertIntoContainer (TcVnVector4_USINT) [▶ 2171]

InsertIntoContainer (ULONG) [▶ 2171]

InsertIntoContainer (USHORT) [▶ 2172]

InsertIntoContainer (ULONGLONG) [▶ 2173]

InsertIntoContainer (unsigned char) [▶ 2174]

InverseDft [▶ 2422]

InvertAffineTransformation [▶ 2480]

InvertImageColor [▶ 2536]

InvertMatrix3x3 [▶ 2692]

IteratorDistance [▶ 2221]

LaplacianFilter [▶ 2557]

LineIntersectionPoint [▶ 2693]

LineIntersectionPointAndAngle [▶ 2694]

LocalMaxima [▶ 2558]

LocalMinima [▶ 2559]

MaxContainer [▶ 1999]

MaxElement [▶ 2346]

MaxElementElementwise (LONG) [▶ 2276]

MaxElementElementwise (SHORT) [▶ 2277]

MaxElementElementwise (double) [▶ 2278]

MaxElementElementwise (float) [▶ 2278]

MaxElementElementwise (char) [▶ 2279]

MaxElementElementwise (TcVnPoint2_DINT) [▶ 2280]

MaxElementElementwise (TcVnPoint2_LREAL) [▶ 2280]

MaxElementElementwise (TcVnPoint2_REAL) [▶ 2281]

MaxElementElementwise (TcVnPoint3_LREAL) [▶ 2282]

MaxElementElementwise (TcVnPoint3_REAL) [▶ 2282]

MaxElementElementwise (TcVnVector2_DINT) [▶ 2283]

MaxElementElementwise (TcVnVector2_INT) [▶ 2284]

MaxElementElementwise (TcVnVector2_REAL) [▶ 2284]

MaxElementElementwise (TcVnVector2_SINT) [▶ 2285]

MaxElementElementwise (TcVnVector2_UINT) [▶ 2286]

MaxElementElementwise (TcVnVector2_USINT) [▶ 2286]

MaxElementElementwise (TcVnVector3_INT) [▶ 2287]

MaxElementElementwise (TcVnVector3_REAL) [▶ 2288]

MaxElementElementwise (TcVnVector3_SINT) [▶ 2288]

MaxElementElementwise (TcVnVector3_UINT) [▶ 2289]

MaxElementElementwise (TcVnVector3_USINT) [▶ 2290]

MaxElementElementwise (TcVnVector4_DINT) [▶ 2290]

MaxElementElementwise (TcVnVector4_INT) [▶ 2291]

MaxElementElementwise (TcVnVector4_LREAL) [▶ 2292]

MaxElementElementwise (TcVnVector4_REAL) [▶ 2297]

MaxElementElementwise (TcVnVector4_SINT) [▶ 2292]

MaxElementElementwise (TcVnVector4_UINT) [▶ 2293]

MaxElementElementwise (TcVnVector4_USINT) [▶ 2294]

MaxElementElementwise (ULONG) [▶ 2294]

MaxElementElementwise (USHORT) [▶ 2295]

MaxElementElementwise (ULONGLONG) [▶ 2296]

MaxElementElementwise (unsigned char) [▶ 2296]

MaxImage [▶ 2026]

MaxImage (Exp) [▶ 2027]

MaxImageWithScalar [▶ 2028]

MaxImageWithScalar (Exp) [▶ 2028]

MaxImageWithVector [▶ 2029]

MaxImageWithVector (Exp) [▶ 2030]

MaxPixelValue [▶ 2523]

MedianElement [▶ 2347]

MedianElementElementwise (LONG) [▶ 2298]

MedianElementElementwise (SHORT) [▶ 2298]

MedianElementElementwise (double) [▶ 2299]

MedianElementElementwise (float) [▶ 2300]

MedianElementElementwise (char) [▶ 2300]

MedianElementElementwise (TcVnPoint2_DINT) [▶ 2301]

MedianElementElementwise (TcVnPoint2_LREAL) [▶ 2302]

MedianElementElementwise (TcVnPoint2_REAL) [▶ 2302]

MedianElementElementwise (TcVnPoint3_LREAL) [▶ 2303]

MedianElementElementwise (TcVnPoint3_REAL) [▶ 2304]

MedianElementElementwise (TcVnVector2_DINT) [▶ 2304]

MedianElementElementwise (TcVnVector2_INT) [▶ 2305]

MedianElementElementwise (TcVnVector2_REAL) [▶ 2306]

MedianElementElementwise (TcVnVector2_SINT) [▶ 2306]

MedianElementElementwise (TcVnVector2_UINT) [▶ 2307]

MedianElementElementwise (TcVnVector2_USINT) [▶ 2308]

MedianElementElementwise (TcVnVector3_INT) [▶ 2308]

MedianElementElementwise (TcVnVector3_REAL) [▶ 2309]

MedianElementElementwise (TcVnVector3_SINT) [▶ 2310]

MedianElementElementwise (TcVnVector3_UINT) [▶ 2310]

MedianElementElementwise (TcVnVector3_USINT) [▶ 2311]

MedianElementElementwise (TcVnVector4_DINT) [▶ 2312]

MedianElementElementwise (TcVnVector4_INT) [▶ 2312]

MedianElementElementwise (TcVnVector4_LREAL) [▶ 2313]

MedianElementElementwise (TcVnVector4_REAL) [▶ 2318]

MedianElementElementwise (TcVnVector4_SINT) [▶ 2314]

MedianElementElementwise (TcVnVector4_UINT) [▶ 2314]

MedianElementElementwise (TcVnVector4_USINT) [▶ 2315]

MedianElementElementwise (ULONG) [▶ 2316]

MedianElementElementwise (USHORT) [▶ 2316]

MedianElementElementwise (ULONGLONG) [▶ 2317]

MedianElementElementwise (unsigned char) [▶ 2318]

MedianFilter [▶ 2560]

MinContainer [▶ 2000]

MinElement [▶ 2348]

MinElementElementwise (LONG) [▶ 2319]

MinElementElementwise (SHORT) [▶ 2320]

MinElementElementwise (double) [▶ 2320]

MinElementElementwise (float) [▶ 2321]

MinElementElementwise (char) [▶ 2322]

MinElementElementwise (TcVnPoint2_DINT) [▶ 2322]

MinElementElementwise (TcVnPoint2_LREAL) [▶ 2323]

MinElementElementwise (TcVnPoint2_REAL) [▶ 2324]

MinElementElementwise (TcVnPoint3_LREAL) [▶ 2324]

MinElementElementwise (TcVnPoint3_REAL) [▶ 2325]

MinElementElementwise (TcVnVector2_DINT) [▶ 2326]

MinElementElementwise (TcVnVector2_INT) [▶ 2326]

MinElementElementwise (TcVnVector2_REAL) [▶ 2327]

MinElementElementwise (TcVnVector2_SINT) [▶ 2328]

MinElementElementwise (TcVnVector2_UINT) [▶ 2328]

MinElementElementwise (TcVnVector2_USINT) [▶ 2329]

MinElementElementwise (TcVnVector3_INT) [▶ 2330]

MinElementElementwise (TcVnVector3_REAL) [▶ 2330]

MinElementElementwise (TcVnVector3_SINT) [▶ 2331]

MinElementElementwise (TcVnVector3_UINT) [▶ 2332]

MinElementElementwise (TcVnVector3_USINT) [▶ 2332]

MinElementElementwise (TcVnVector4_DINT) [▶ 2333]

MinElementElementwise (TcVnVector4_INT) [▶ 2334]

MinElementElementwise (TcVnVector4_LREAL) [▶ 2334]

MinElementElementwise (TcVnVector4_REAL) [▶ 2340]

MinElementElementwise (TcVnVector4_SINT) [▶ 2335]

MinElementElementwise (TcVnVector4_UINT) [▶ 2336]

MinElementElementwise (TcVnVector4_USINT) [▶ 2336]

MinElementElementwise (ULONG) [▶ 2337]

MinElementElementwise (USHORT) [▶ 2338]

MinElementElementwise (ULONGLONG) [▶ 2338]

MinElementElementwise (unsigned char) [▶ 2339]

MinImage [▶ 2031]

MinImage (Exp) [▶ 2032]

MinImageWithScalar [▶ 2033]

MinImageWithScalar (Exp) [▶ 2033]

MinImageWithVector [▶ 2034]

MinImageWithVector (Exp) [▶ 2035]

MinPixelValue [▶ 2524]

MixImageChannels [▶ 2245]

MorphologicalOperator [▶ 2560]

MultiplyContainers [▶ 2001]

MultiplyImages [▶ 2036]

MultiplyImageWithScalar [▶ 2036]

MultiplyImageWithVector [▶ 2037]

MultiplyMatrices [▶ 2695]

MultiplyWithContainerElements (1) [▶ 2001]

MultiplyWithContainerElements (2) [▶ 2002]

MultiplyWithContainerElements (3) [▶ 2003]

MultiplyWithContainerElements (4) [▶ 2003]

NegateContainer [▶ 2004]

NonMaxSuppression [▶ 2720]

NormalizeImage [▶ 2536]

NormalizeImageForDisplay [▶ 2538]

OptimalDftSize [▶ 2423]

OrthogonalVector (TcVnVector2_DINT) [▶ 2699]

OrthogonalVector (TcVnVector2_LREAL) [▶ 2699]

OrthogonalVector (TcVnVector2_REAL) [▶ 2700]

PadImageBorder [▶ 2256]

PerspectiveTransformation [▶ 2481]

PlotIntensityProfile [▶ 2408]

PointToLineDistance (TcVnPoint2_DINT) [▶ 2701]

PointToLineDistance (TcVnPoint2_LREAL) [▶ 2702]

PointToLineDistance (TcVnPoint2_REAL) [▶ 2703]

PointToPointDirection (TcVnPoint2_DINT) [▶ 2703]

PointToPointDirection (TcVnPoint2_LREAL) [▶ 2704]

PointToPointDirection (TcVnPoint2_REAL) [▶ 2705]

PointToPointDistance (TcVnPoint2_DINT) [▶ 2706]

PointToPointDistance (TcVnPoint2_LREAL) [▶ 2707]

PointToPointDistance (TcVnPoint2_REAL) [▶ 2707]

PutLabel [▶ 2410]

PutText [▶ 2411]

PyramidDown [▶ 2482]

PyramidUp [▶ 2483]

Reduce [▶ 2722]

ReduceArg [▶ 2723]

ReferenceColorSimilarity (ITcVnColorModel*) [▶ 2538]

ReferenceColorSimilarity (ITcVnMlModel*) [▶ 2539]

ReferenceColorSimilarity (TcVnVector3_LREAL) [▶ 2540]

RegionOrientation [▶ 2525]

ReinterpretUnsupportedImage [▶ 2257]

RemapImageToLogPolarSpace [▶ 2483]

RemapImageToPolarSpace [▶ 2485]

RemoveLocalMaxima [▶ 2561]

RemoveLocalMinima [▶ 2562]

ReserveContainerMemory [▶ 2222]

ResetRoi [▶ 2246]

ReshapeImage [▶ 2723]

ResizeImage [▶ 2486]

ReverseContainer [▶ 2222]

RotatedRectangleCorners [▶ 2696]

RotatedRectangleIntersection [▶ 2697]

RotateImage [▶ 2487]

RotateImage (Exp) [▶ 2487]

SauvolaThreshold [▶ 2573]

ScharrFilter [▶ 2563]

SeparableCustomFilter [▶ 2564]

SetAt (LONG) [▶ 2176]

SetAt (SHORT) [▶ 2177]

SetAt (ITcVnContainer*) [▶ 2178]

SetAt (ITcVnImage*) [▶ 2178]

SetAt (double) [▶ 2179]

SetAt (float) [▶ 2180]

SetAt (char) [▶ 2181]

SetAt (TcVnDMatch) [▶ 2181]

SetAt (TcVnKeyPoint) [▶ 2182]

SetAt (TcVnPoint2_DINT) [▶ 2183]

SetAt (TcVnPoint2_LREAL) [▶ 2184]

SetAt (TcVnPoint2_REAL) [▶ 2185]

SetAt (TcVnPoint3_LREAL) [▶ 2185]

SetAt (TcVnPoint3_REAL) [▶ 2186]

SetAt (TcVnRectangle_DINT) [▶ 2187]

SetAt (TcVnRotatedRectangle) [▶ 2204]

SetAt (TcVnVector2_DINT) [▶ 2188]

SetAt (TcVnVector2_INT) [▶ 2188]

SetAt (TcVnVector2_REAL) [▶ 2189]

SetAt (TcVnVector2_SINT) [▶ 2190]

SetAt (TcVnVector2_UINT) [▶ 2191]

SetAt (TcVnVector2_USINT) [▶ 2191]

SetAt (TcVnVector3_INT) [▶ 2192]

SetAt (TcVnVector3_REAL) [▶ 2193]

SetAt (TcVnVector3_SINT) [▶ 2194]

SetAt (TcVnVector3_UINT) [▶ 2194]

SetAt (TcVnVector3_USINT) [▶ 2195]

SetAt (TcVnVector4_DINT) [▶ 2196]

SetAt (TcVnVector4_INT) [▶ 2197]

SetAt (TcVnVector4_LREAL) [▶ 2197]

SetAt (TcVnVector4_REAL) [▶ 2203]

SetAt (TcVnVector4_SINT) [▶ 2198]

SetAt (TcVnVector4_UINT) [▶ 2199]

SetAt (TcVnVector4_USINT) [▶ 2200]

SetAt (ULONG) [▶ 2200]

SetAt (USHORT) [▶ 2201]

SetAt (ULONGLONG) [▶ 2202]

SetAt (unsigned char) [▶ 2202]

SetContainer [▶ 2223]

SetImageChannel [▶ 2247]

SetIteratorToBegin [▶ 2224]

SetPixel [▶ 2247]

SetPixels [▶ 2248]

SetRngSeed [▶ 2697]

SetRoi [▶ 2249]

SetRoi (TcVnRectangle_DINT) [▶ 2250]

SetRoi (TcVnRectangle_UDINT) [▶ 2250]

Sigmoid [▶ 2724]

SobelFilter [▶ 2565]

SoftMax [▶ 2725]

SplitImageChannels [▶ 2251]

StartAbsWatchdog [▶ 2377]

StartRelWatchdog [▶ 2377]

StopWatchdog [▶ 2378]

SubtractContainers [▶ 2005]

SubtractImageFromScalar [▶ 2038]

SubtractImageFromVector [▶ 2038]

SubtractImages [▶ 2039]

SubtractScalarFromImage [▶ 2040]

SubtractVectorFromImage [▶ 2041]

Threshold [▶ 2571]

TrainImageColor [▶ 2542]

TrainImageColor (ITcVnMlModel*) [▶ 2545]

TransformCoordinatesPlanar (Container) [▶ 2453]

TransformCoordinatesPlanar (Point) [▶ 2454]

TransformIntoDisplayableImage [▶ 2252]

UpdateTimestamp [▶ 2698]

UprightBoundingRectangle [▶ 2376]

UprightInnerRectangle [▶ 2527]

VarianceFilter [▶ 2566]

WarpAffine [▶ 2488]

WarpAffine (Container) [▶ 2489]

WarpAffine (Point) [▶ 2490]

WarpAffine (Rectangle) [▶ 2491]

WarpAffine (Exp) [▶ 2492]

WarpPerspective [▶ 2493]

WarpPerspective (Container) [▶ 2493]

WarpPerspective (Point) [▶ 2494]

WarpPerspective (Rectangle) [▶ 2495]

WarpPerspective (Exp) [▶ 2496]

WatershedSegmentation (Exp) [▶ 2572]

WhiteBalance [▶ 2548]

TC3 Vision Code Reading

ReadBarcode [▶ 2267]

ReadBarcode (Exp) [▶ 2268]

ReadDataMatrixCode [▶ 2270]

ReadDataMatrixCodeRoi [▶ 2274]

ReadPharmaCode [▶ 2271]

ReadQRCode [▶ 2272]

TC3 Vision Code Quality

GradeBarcode [▶ 2259]

GradeDataMatrixCode [▶ 2261]

GradeQRCode [▶ 2263]

TC3 Vision Metrology 2D

AdjustSearchWindowOrientationToLinearEdge [▶ 2656]

CalibrateCamera [▶ 2426]

CalibrateCamera (Exp) [▶ 2427]

CalibrateCameraManually [▶ 2431]

CalibrateLinescanCamera [▶ 2435]

ClosestPointsBF [▶ 2657]

CompensateLensDistortion [▶ 2438]

CompensateLensDistortion (Exp1) [▶ 2439]

CompensateLensDistortion (Exp2) [▶ 2440]

CompensateLensDistortionForPoints [▶ 2441]

CompensateLensDistortionForPoints (Exp1) [▶ 2442]

CompensateLensDistortionForPoints (Exp2) [▶ 2443]

DetectPatternPoints [▶ 2444]

GenerateCalibrationPatternReferencePoints [▶ 2447]

ImagePointsWorldDistance [▶ 2448]

LocateCircularArc [▶ 2658]

LocateEdge [▶ 2661]

LocateEdges [▶ 2665]

LocateEllipse [▶ 2668]

MeasureAngleBetweenEdges [▶ 2671]

MeasureEdgeDistance [▶ 2674]

MeasureMinEdgeDistance [▶ 2678]

SolvePnP [▶ 2449]

SortDetectedPatternPoints [▶ 2451]

TransformCoordinatesImageToWorld (Container) [▶ 2451]

TransformCoordinatesImageToWorld (Point) [▶ 2452]

TransformCoordinatesWorldToImage (Container) [▶ 2455]

TransformCoordinatesWorldToImage (Point) [▶ 2456]

TC3 Vision Matching

DrawKeypoints [▶ 2388]

DrawMatches [▶ 2394]

FilterGoodMatches [▶ 2575]

FindReferenceKeyPointsInImage [▶ 2576]

FindReferenceKeyPointsInImageAKAZE [▶ 2580]

FindReferenceKeyPointsInImageBRISK [▶ 2584]

FindReferenceKeyPointsInImageORB [▶ 2588]

GetMatchCoordinates [▶ 2592]

KeyPointsAGAST [▶ 2593]

KeyPointsAndDescriptorsAKAZE [▶ 2594]

KeyPointsAndDescriptorsBRISK [▶ 2595]

KeyPointsAndDescriptorsKAZE [▶ 2596]

KeyPointsAndDescriptorsORB [▶ 2597]

KeyPointsFAST [▶ 2598]

KeyPointsGFTT [▶ 2599]

KeyPointsMSER [▶ 2600]

KeyPointsSB [▶ 2601]

MatchContours [▶ 2372]

MatchContours1vsN [▶ 2374]

MatchDescriptorsBF [▶ 2602]

MatchDescriptorsFlannLsh [▶ 2603]

MatchDescriptorsKnnBF [▶ 2604]

MatchDescriptorsKnnFlannLsh [▶ 2607]

MatchImageHuMoments [▶ 2508]

MatchTemplate [▶ 2508]

MatchTemplateAndEvaluate [▶ 2510]

RegionsMSER [▶ 2608]

TC3 Vision OCR

OCR [▶ 2728]

TC3 Machine Learning Realtime Inference

CreateBoostClassifier [▶ 2610]

CreateKmppModel [▶ 2611]

CreateKnnModel [▶ 2612]

CreateLbgModel [▶ 2613]

CreateLdaTransform [▶ 2615]

CreateLdaTransformViaComponentNum [▶ 2616]

CreateNbcModel [▶ 2617]

CreatePcaTransform [▶ 2617]

CreatePcaTransformViaComponentNum [▶ 2618]

CreatePcaTransformViaVariance [▶ 2619]

CreateRTreesModel [▶ 2620]

CreateStaModel [▶ 2622]

CreateSvmModel [▶ 2625]

CreateSvmSgdClassifier [▶ 2628]

FeatureScaling [▶ 2630]

FeatureTransform [▶ 2631]

GetBatchClusters [▶ 2632]

GetBatchNovelties [▶ 2632]

GetClusterCenter [▶ 2633]

GetClusterNum [▶ 2634]

GetFeatureScales [▶ 2635]

GetSampleCluster [▶ 2636]

GetSampleNovelty [▶ 2636]

Granulometry [▶ 2730]

HaralickFeatures [▶ 2731]

InverseFeatureScaling [▶ 2637]

InverseFeatureScaling (float) [▶ 2639]

InverseFeatureTransform [▶ 2639]

PredictBatch [▶ 2640]

PredictSampleClass [▶ 2641]

PredictSampleScalar [▶ 2642]

PredictSampleVector [▶ 2643]

TrainBatch [▶ 2644]

TrainBatchClusters [▶ 2645]

TrainSample [▶ 2646]

TrainSampleClass [▶ 2647]

TrainSampleCluster [▶ 2648]

TrainSampleScalar [▶ 2648]

TrainSampleVector [▶ 2649]

TC3 Neural Network Realtime Inference

ExecuteNeuralNetwork [▶ 2710]

ExecuteNeuralNetwork (MI) [▶ 2711]

ExecuteNeuralNetwork (MIMO_String) [▶ 2712]

ExecuteNeuralNetwork (MISO) [▶ 2713]

ExecuteNeuralNetwork (MO) [▶ 2714]

ExecuteNeuralNetwork_MultiLayerOutput [▶ 2716]

ExecuteNeuralNetwork_SingleLayerOutput [▶ 2717]

ExecuteNeuralNetwork (SO) [▶ 2715]

GetNeuralNetworkLayerNames [▶ 2718]

GetNeuralNetworkLayerNames (Exp) [▶ 2718]

GetNeuralNetworkLayerNames (String) [▶ 2719]

6.2.3.2 Algebraic Container Operations

Diese Gruppe enthält Funktionen zur elementweisen Anwendung von algebraischen Operatoren auf Containern [▶ 163].

Funktionen

Algebra

- AddContainers [▶ 1995]

- AddToContainerElements [▶ 1972]
- DivideContainers [▶ 1999]
- MultiplyContainers [▶ 2001]
- MultiplyWithContainerElements [▶ 2001]
- NegateContainer [▶ 2004]
- SubtractContainers [▶ 2005]

Bitweise Operatoren

- BitwiseAndContainers [▶ 1996]
- BitwiseNotContainer [▶ 1997]
- BitwiseOrContainers [▶ 1997]
- BitwiseXorContainers [▶ 1998]

Statistische Merkmale

- MaxContainer [▶ 1999]
- MinContainer [▶ 2000]

6.2.3.2.1 AddToContainerElements

Dieses Kapitel enthält Funktionen, mit denen ein Wert zu jedem einzelnen Container-Element entsprechend dem Datentyp addiert werden kann.

6.2.3.2.1.1 AddToContainerElements (LONG)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT         hrPrev,
    LONG            nValue,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nValue	LONG	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with DINT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.2 AddToContainerElements (SHORT)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT         hrPrev,
    SHORT           nValue,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nValue	SHORT	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with INT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.3 AddToContainerElements (double)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT         hrPrev,
    double          fValue,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fValue	double	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with LREAL elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.4 AddToContainerElements (float)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT         hrPrev,
    float           fValue,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fValue	float	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with REAL elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.5 AddToContainerElements (char)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT         hrPrev,
    char            nValue,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nValue	char	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with SINT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.6 AddToContainerElements (TcVnKeyPoint)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT         hrPrev,
    TcVnKeyPoint&   stValue,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stValue	TcVnKeyPoint [▶ 1768]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnKeyPoint elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.7 AddToContainerElements (TcVnPoint2_DINT)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT          hrPrev,
    TcVnPoint2_DINT& aValue,
    ITcVnContainer*  ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnPoint2_DINT [▶ 1711]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnPoint2_DINT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.8 AddToContainerElements (TcVnPoint2_LREAL)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT           hrPrev,
    TcVnPoint2_LREAL& aValue,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnPoint2_LREAL [▶ 1711]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnPoint2_LREAL elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.9 AddToContainerElements (TcVnPoint2_REAL)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT          hrPrev,
    TcVnPoint2_REAL& aValue,
    ITcVnContainer*  ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnPoint2_REAL [▶ 1711]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnPoint2_REAL elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.10 AddToContainerElements (TcVnPoint3_LREAL)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT           hrPrev,
    TcVnPoint3_LREAL& aValue,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnPoint3_LREAL [▶ 1711]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnPoint3_LREAL elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.11 AddToContainerElements (TcVnPoint3_REAL)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT          hrPrev,
    TcVnPoint3_REAL& aValue,
    ITcVnContainer*  ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnPoint3_REAL [▶ 1711]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnPoint3_REAL elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.12 AddToContainerElements (TcVnRectangle_DINT)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT             hrPrev,
    TcVnRectangle_DINT& stValue,
    ITcVnContainer*     ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stValue	TcVnRectangle_DINT [▶ 1781]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnRectangle_DINT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.13 AddToContainerElements (TcVnVector2_DINT)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT           hrPrev,
    TcVnVector2_DINT& aValue,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector2_DINT [▶ 1713]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector2_DINT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.14 AddToContainerElements (TcVnVector2_INT)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT          hrPrev,
    TcVnVector2_INT& aValue,
    ITcVnContainer*  ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector2_INT [▶ 1713]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector2_INT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.15 AddToContainerElements (TcVnVector2_REAL)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT           hrPrev,
    TcVnVector2_REAL& aValue,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector2_REAL [▶ 1713]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector2_REAL elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.16 AddToContainerElements (TcVnVector2_SINT)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT           hrPrev,
    TcVnVector2_SINT& aValue,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector2_SINT [▶ 1713]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector2_SINT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.17 AddToContainerElements (TcVnVector2_UINT)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT           hrPrev,
    TcVnVector2_UINT& aValue,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector2_UINT [▶ 1713]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector2_UINT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.18 AddToContainerElements (TcVnVector2_USINT)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT            hrPrev,
    TcVnVector2_USINT& aValue,
    ITcVnContainer*    ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector2_USINT [▶ 1713]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector2_USINT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.19 AddToContainerElements (TcVnVector3_INT)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT          hrPrev,
    TcVnVector3_INT& aValue,
    ITcVnContainer*  ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector3_INT [▶ 1713]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector3_INT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.20 AddToContainerElements (TcVnVector3_REAL)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT           hrPrev,
    TcVnVector3_REAL& aValue,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector3_REAL [▶ 1713]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector3_REAL elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.21 AddToContainerElements (TcVnVector3_SINT)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT           hrPrev,
    TcVnVector3_SINT& aValue,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector3_SINT [▶ 1713]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector3_SINT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.22 AddToContainerElements (TcVnVector3_UINT)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT           hrPrev,
    TcVnVector3_UINT& aValue,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector3_UINT [▶ 1713]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector3_UINT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.23 AddToContainerElements (TcVnVector3_USINT)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT            hrPrev,
    TcVnVector3_USINT& aValue,
    ITcVnContainer*    ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector3_USINT [▶ 1713]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector3_USINT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.24 AddToContainerElements (TcVnVector4_DINT)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT           hrPrev,
    TcVnVector4_DINT& aValue,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector4_DINT [▶ 1713]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector4_DINT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.25 AddToContainerElements (TcVnVector4_INT)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT          hrPrev,
    TcVnVector4_INT& aValue,
    ITcVnContainer*  ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector4_INT [▶ 1713]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector4_INT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.26 AddToContainerElements (TcVnVector4_LREAL)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT            hrPrev,
    TcVnVector4_LREAL& aValue,
    ITcVnContainer*    ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector4_LREAL [▶ 1713]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector4_LREAL elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.27 AddToContainerElements (TcVnVector4_SINT)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT           hrPrev,
    TcVnVector4_SINT& aValue,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector4_SINT [▶ 1713]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector4_SINT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.28 AddToContainerElements (TcVnVector4_UINT)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT           hrPrev,
    TcVnVector4_UINT& aValue,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector4_UINT [▶ 1713]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector4_UINT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.29 AddToContainerElements (TcVnVector4_USINT)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT            hrPrev,
    TcVnVector4_USINT& aValue,
    ITcVnContainer*    ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector4_USINT [▶ 1713]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector4_USINT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.30 AddToContainerElements (ULONG)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT         hrPrev,
    ULONG           nValue,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nValue	ULONG	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with UDINT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.31 AddToContainerElements (USHORT)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT         hrPrev,
    USHORT          nValue,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nValue	USHORT	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with UINT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.32 AddToContainerElements (ULONGLONG)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT         hrPrev,
    ULONGLONG       nValue,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nValue	ULONGLONG	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with ULINT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.33 AddToContainerElements (unsigned char)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT         hrPrev,
    unsigned char   nValue,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nValue	unsigned char	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with USINT elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.34 AddToContainerElements (TcVnVector4_REAL)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT           hrPrev,
    TcVnVector4_REAL& aValue,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector4_REAL [▶ 1713]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector4_REAL elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.1.35 AddToContainerElements (TcVnRotatedRectangle)

Add a value to each container element.

Syntax

Definition:

```
HRESULT AddToContainerElements(
    HRESULT               hrPrev,
    TcVnRotatedRectangle& stValue,
    ITcVnContainer*       ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stValue	TcVnRotatedRectangle [▶ 1782]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnRotatedRectangle elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.2 AddContainers

Element-wise addition of two containers (same length and type).

Syntax

Definition:

```
HRESULT AddContainers(
    HRESULT          hrPrev,
    ITcVnContainer*  ipSrcContainer1,
    ITcVnContainer*  ipSrcContainer2,
    ITcVnContainer*& ipDestContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer1	ITcVnContainer* [▶ 1890]	Source container 1
ipSrcContainer2	ITcVnContainer* [▶ 1890]	Source container 2
ipDestContainer	ITcVnContainer* [▶ 1890]&	Returns the resulting container

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.3 BitwiseAndContainers

Element-wise application of a bit-wise AND operator to two containers (same length and type, integer only).

Syntax

Definition:

```
HRESULT BitwiseAndContainers(
    HRESULT          hrPrev,
    ITcVnContainer*  ipSrcContainer1,
    ITcVnContainer*  ipSrcContainer2,
    ITcVnContainer*& ipDestContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer1	ITcVnContainer* [▶ 1890]	Source container 1
ipSrcContainer2	ITcVnContainer* [▶ 1890]	Source container 2
ipDestContainer	ITcVnContainer* [▶ 1890]&	Returns the resulting container

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.4 BitwiseNotContainer

Element-wise application of a bit-wise NOT operator to a container (integer only).

Syntax

Definition:

```
HRESULT BitwiseNotContainer(
    HRESULT          hrPrev,
    ITcVnContainer*  ipSrcContainer,
    ITcVnContainer*& ipDestContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1890]	Source container
ipDestContainer	ITcVnContainer* [▶ 1890]&	Returns the resulting container

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.5 BitwiseOrContainers

Element-wise application of a bit-wise OR operator to two containers (same length and type, integer only).

Syntax

Definition:

```
HRESULT BitwiseOrContainers(
    HRESULT          hrPrev,
    ITcVnContainer*  ipSrcContainer1,
    ITcVnContainer*  ipSrcContainer2,
    ITcVnContainer*& ipDestContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer1	ITcVnContainer* [▶ 1890]	Source container 1
ipSrcContainer2	ITcVnContainer* [▶ 1890]	Source container 2
ipDestContainer	ITcVnContainer* [▶ 1890]&	Returns the resulting container

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.6 BitwiseXorContainers

Element-wise application of a bit-wise XOR operator to two containers (same length and type, integer only).

Syntax

Definition:

```
HRESULT BitwiseXorContainers(
    HRESULT          hrPrev,
    ITcVnContainer*  ipSrcContainer1,
    ITcVnContainer*  ipSrcContainer2,
    ITcVnContainer*& ipDestContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer1	ITcVnContainer* [▶ 1890]	Source container 1
ipSrcContainer2	ITcVnContainer* [▶ 1890]	Source container 2
ipDestContainer	ITcVnContainer* [▶ 1890]&	Returns the resulting container

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.7 DivideContainers

Element-wise division of two containers (same length and type). In case of integer division by zero, the resulting element is set to nIntDivideByZeroResult and S_DIVISION_BY_ZERO is returned instead of S_OK.

Syntax

Definition:

```
HRESULT DivideContainers(
    HRESULT          hrPrev,
    ITcVnContainer*  ipSrcContainer1,
    ITcVnContainer*  ipSrcContainer2,
    ITcVnContainer*& ipDestContainer,
    LONGLONG         nIntDivideByZeroResult = 0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer1	ITcVnContainer* [▶ 1890]		Source container 1
ipSrcContainer2	ITcVnContainer* [▶ 1890]		Source container 2
ipDestContainer	ITcVnContainer* [▶ 1890]&		Returns the resulting container
nIntDivideByZeroResult	LONGLONG	0	Value set as result in case of integer division by zero

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.8 MaxContainer

Element-wise maximum of two containers (same length and type).

Syntax

Definition:

```
HRESULT MaxContainer(
    HRESULT          hrPrev,
    ITcVnContainer*  ipSrcContainer1,
    ITcVnContainer*  ipSrcContainer2,
    ITcVnContainer*& ipDestContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer1	ITcVnContainer* [▶ 1890]	Source container 1
ipSrcContainer2	ITcVnContainer* [▶ 1890]	Source container 2
ipDestContainer	ITcVnContainer* [▶ 1890]&	Returns the resulting container

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.9 MinContainer

Element-wise minimum of two containers (same length and type).

Syntax

Definition:

```
HRESULT MinContainer(
    HRESULT          hrPrev,
    ITcVnContainer*  ipSrcContainer1,
    ITcVnContainer*  ipSrcContainer2,
    ITcVnContainer*& ipDestContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer1	ITcVnContainer* [▶ 1890]	Source container 1
ipSrcContainer2	ITcVnContainer* [▶ 1890]	Source container 2
ipDestContainer	ITcVnContainer* [▶ 1890]&	Returns the resulting container

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.10 MultiplyContainers

Element-wise multiplication of two containers (same length and type).

Syntax

Definition:

```
HRESULT MultiplyContainers(
    HRESULT          hrPrev,
    ITcVnContainer*  ipSrcContainer1,
    ITcVnContainer*  ipSrcContainer2,
    ITcVnContainer*& ipDestContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer1	ITcVnContainer* [▶ 1890]	Source container 1
ipSrcContainer2	ITcVnContainer* [▶ 1890]	Source container 2
ipDestContainer	ITcVnContainer* [▶ 1890]&	Returns the resulting container

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.11 MultiplyWithContainerElements (1)

Multiply each container element with a value.

Syntax

Definition:

```
HRESULT MultiplyWithContainerElements(
    HRESULT         hrPrev,
    double          fValue,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fValue	double	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with 1-dimensional elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.12 MultiplyWithContainerElements (2)

Multiply each container element with a value.

Syntax

Definition:

```
HRESULT MultiplyWithContainerElements(
    HRESULT            hrPrev,
    TcVnVector2_LREAL& aValue,
    ITcVnContainer*    ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector2_LREAL [▶ 1713]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with 2-dimensional elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.13 MultiplyWithContainerElements (3)

Multiply each container element with a value.

Syntax

Definition:

```
HRESULT MultiplyWithContainerElements(
    HRESULT            hrPrev,
    TcVnVector3_LREAL& aValue,
    ITcVnContainer*    ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector3_LREAL [▶ 1713]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with 3-dimensional elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.14 MultiplyWithContainerElements (4)

Multiply each container element with a value.

Syntax

Definition:

```
HRESULT MultiplyWithContainerElements(
    HRESULT            hrPrev,
    TcVnVector4_LREAL& aValue,
    ITcVnContainer*    ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aValue	TcVnVector4_LREAL [▶ 1713]&	Value
ipContainer	ITcVnContainer* [▶ 1890]	Container with 4-dimensional elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.15 NegateContainer

Element-wise negation of a container (two's complement).

Syntax

Definition:

```
HRESULT NegateContainer(
    HRESULT          hrPrev,
    ITcVnContainer*  ipSrcContainer,
    ITcVnContainer*& ipDestContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1890]	Source container
ipDestContainer	ITcVnContainer* [▶ 1890]&	Returns the resulting container (same type as ipSrcContainer)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.2.16 SubtractContainers

Element-wise subtraction of two containers (same length and type).

Syntax

Definition:

```
HRESULT SubtractContainers(
    HRESULT          hrPrev,
    ITcVnContainer*  ipSrcContainer1,
    ITcVnContainer*  ipSrcContainer2,
    ITcVnContainer*& ipDestContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer1	ITcVnContainer* [▶ 1890]	Source container 1
ipSrcContainer2	ITcVnContainer* [▶ 1890]	Source container 2
ipDestContainer	ITcVnContainer* [▶ 1890]&	Returns the resulting container

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3 Algebraic Image Operations

Diese Gruppe enthält Funktionen zur pixelweisen Anwendung von algebraischen Operationen auf Bildern [▶ 159].

Addition

- AddImages [▶ 2007]
- AddImagesWeighted [▶ 2007]
- AddScalarToImage [▶ 2008]

- AddVectorToImage [▶ 2009]

Subtraktion

- SubtractImages [▶ 2039]
- SubtractImageFromScalar [▶ 2038]
- SubtractImageFromVector [▶ 2038]
- SubtractScalarFromImage [▶ 2040]
- SubtractVectorFromImage [▶ 2041]

Multiplikation

- MultiplyImages [▶ 2036]
- MultiplyImageWithScalar [▶ 2036]
- MultiplyImageWithVector [▶ 2037]

Division

- DivideImages [▶ 2023]
- DivideImageByScalar [▶ 2021]
- DivideImageByVector [▶ 2022]
- DivideScalarByImage [▶ 2023]
- DivideVectorByImage [▶ 2024]

Potenzieren und Logarithmieren

- ElementwiseExp [▶ 2025]
- ElementwiseLog [▶ 2026]

Bitweise AND

- BitwiseAndImages(Exp) [▶ 2010]
- BitwiseAndScalarWithImage [▶ 2012]
- BitwiseAndVectorWithImage [▶ 2012]

Bitweise NOT

- BitwiseNotImage(Exp) [▶ 2013]

Bitweise OR

- BitwiseOrImages(Exp) [▶ 2014]
- BitwiseOrScalarWithImage [▶ 2016]
- BitwiseOrVectorWithImage [▶ 2017]

Bitweise XOR

- BitwiseXorImages(Exp) [▶ 2018]
- BitwiseXorScalarWithImage [▶ 2019]
- BitwiseXorVectorWithImage [▶ 2020]

Überblendung

- BlendImages [▶ 2021]

Maximum

- MaxImage(Exp) [▶ 2026]
- MaxImageWithScalar(Exp) [▶ 2028]

- MaxImageWithVector(Exp) [▶ 2029]

Minimum

- MinImage(Exp) [▶ 2031]
- MinImageWithScalar(Exp) [▶ 2033]
- MinImageWithVector(Exp) [▶ 2034]

6.2.3.3.1 AddImages

Element-wise addition of two images using saturation arithmetics.

Syntax

Definition:

```
HRESULT AddImages(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage1,
    ITcVnImage*  ipSrcImage2,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1925]	First source image
ipSrcImage2	ITcVnImage* [▶ 1925]	Second source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Beispiele

- Bildbereiche kopieren [▶ 2843]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.2 AddImagesWeighted

Weighted, element-wise addition of two images using saturation arithmetics.

Syntax

Definition:

```
HRESULT AddImagesWeighted(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage1,
    ITcVnImage*      ipSrcImage2,
    ITcVnImage*&     ipDestImage,
    double           fWeight1,
    double           fWeight2,
    double           fDelta,
    ETcVnElementType eDestType = TCVN_ET_SAME_AS_SOURCE
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1925]		First source image
ipSrcImage2	ITcVnImage* [▶ 1925]		Second source image
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (An appropriate destination image will be created if required.)
fWeight1	double		Weight factor applied to ipSrcImage1
fWeight2	double		Weight factor applied to ipSrcImage2
fDelta	double		Value added to the weighted sum of both images
eDestType	ETcVnElementType [▶ 1738]	TCVN_ET_SAME_AS_SOURCE	Destination image depth

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.3 AddScalarToImage

Add a scalar value to each image pixel using saturation arithmetics.

Syntax

Definition:

```
HRESULT AddScalarToImage(
    HRESULT      hrPrev,
    double       fScalar,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fScalar	double	Scalar value
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.4 AddVectorToImage

Add a vector (1 element for each image channel) to each image pixel using saturation arithmetics.

Syntax

Definition:

```
HRESULT AddVectorToImage(
    HRESULT            hrPrev,
    TcVnVector4_LREAL& aVector,
    ITcVnImage*        ipSrcImage,
    ITcVnImage*&       ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aVector	TcVnVector4_LREAL [▶ 1713]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.5 BitwiseAndImages

Element-wise application of a bit-wise AND operator to two images.

Syntax

Definition:

```
HRESULT BitwiseAndImages(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage1,
    ITcVnImage*  ipSrcImage2,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1925]	First source image
ipSrcImage2	ITcVnImage* [▶ 1925]	Second source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.6 BitwiseAndImages (Exp)

Element-wise application of a bit-wise AND operator to two images.

Syntax

Definition:

```
HRESULT BitwiseAndImages(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage1,
    ITcVnImage*  ipSrcImage2,
    ITcVnImage*& ipDestImage,
    ITcVnImage*  ipMask
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1925]	First source image
ipSrcImage2	ITcVnImage* [▶ 1925]	Second source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage* [▶ 1925]	Mask to limit the operation to specific pixel positions (mask value 0: skip pixel, mask value > 0: apply operation to pixel)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.7 BitwiseAndScalarWithImage

Bitwise and a scalar value with each image pixel.

Syntax

Definition:

```
HRESULT BitwiseAndScalarWithImage(
    HRESULT      hrPrev,
    double       fScalar,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fScalar	double	Scalar value
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.8 BitwiseAndVectorWithImage

Bitwise and a vector (1 element for each image channel) with each image pixel.

Syntax

Definition:

```
HRESULT BitwiseAndVectorWithImage(
    HRESULT            hrPrev,
    TcVnVector4_LREAL& aVector,
    ITcVnImage*        ipSrcImage,
    ITcVnImage*&       ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aVector	TcVnVector4_LREAL [▶ 1713]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.9 BitwiseNotImage

Element-wise application of a bit-wise NOT operator to an image.

Syntax

Definition:

```
HRESULT BitwiseNotImage(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.10 BitwiseNotImage (Exp)

Element-wise application of a bit-wise NOT operator to an image.

Syntax

Definition:

```
HRESULT BitwiseNotImage(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage,
    ITcVnImage*  ipMask
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage* [▶ 1925]	Mask to limit the operation to specific pixel positions (mask value 0: skip pixel, mask value > 0: apply operation to pixel)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.11 BitwiseOrImages

Element-wise application of a bit-wise OR operator to two images.

Syntax

Definition:

```
HRESULT BitwiseOrImages(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage1,
    ITcVnImage*  ipSrcImage2,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1925]	First source image
ipSrcImage2	ITcVnImage* [▶ 1925]	Second source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.12 BitwiseOrImages (Exp)

Element-wise application of a bit-wise OR operator to two images.

Syntax

Definition:

```
HRESULT BitwiseOrImages(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage1,
    ITcVnImage*  ipSrcImage2,
    ITcVnImage*& ipDestImage,
    ITcVnImage*  ipMask
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1925]	First source image
ipSrcImage2	ITcVnImage* [▶ 1925]	Second source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage* [▶ 1925]	Mask to limit the operation to specific pixel positions (mask value 0: skip pixel, mask value > 0: apply operation to pixel)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.13 BitwiseOrScalarWithImage

Bitwise or a scalar value with each image pixel.

Syntax

Definition:

```
HRESULT BitwiseOrScalarWithImage(
    HRESULT      hrPrev,
    double       fScalar,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fScalar	double	Scalar value
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.14 BitwiseOrVectorWithImage

Bitwise or a vector (1 element for each image channel) with each image pixel.

Syntax

Definition:

```
HRESULT BitwiseOrVectorWithImage(
    HRESULT            hrPrev,
    TcVnVector4_LREAL& aVector,
    ITcVnImage*        ipSrcImage,
    ITcVnImage*&       ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aVector	TcVnVector4_LREAL [▶ 1713]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.15 BitwiseXorImages

Element-wise application of a bit-wise XOR operator to two images.

Syntax

Definition:

```
HRESULT BitwiseXorImages(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage1,
    ITcVnImage*  ipSrcImage2,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1925]	First source image
ipSrcImage2	ITcVnImage* [▶ 1925]	Second source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.16 BitwiseXorImages (Exp)

Element-wise application of a bit-wise XOR operator to two images.

Syntax

Definition:

```
HRESULT BitwiseXorImages(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage1,
    ITcVnImage*  ipSrcImage2,
    ITcVnImage*& ipDestImage,
    ITcVnImage*  ipMask
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1925]	First source image
ipSrcImage2	ITcVnImage* [▶ 1925]	Second source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage* [▶ 1925]	Mask to limit the operation to specific pixel positions (mask value 0: skip pixel, mask value > 0: apply operation to pixel)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.17 BitwiseXorScalarWithImage

Bitwise xor a scalar value with each image pixel.

Syntax

Definition:

```
HRESULT BitwiseXorScalarWithImage(
    HRESULT      hrPrev,
    double       fScalar,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fScalar	double	Scalar value
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.18 BitwiseXorVectorWithImage

Bitwise xor a vector (1 element for each image channel) with each image pixel.

Syntax

Definition:

```
HRESULT BitwiseXorVectorWithImage(
    HRESULT            hrPrev,
    TcVnVector4_LREAL& aVector,
    ITcVnImage*        ipSrcImage,
    ITcVnImage*&       ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aVector	TcVnVector4_LREAL [▶ 1713]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.19 BlendImages

Blends two images, i.e. dest = (src1 * weight1 + src2 * weight2) / (weight1 + weight2).
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT BlendImages(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage1,
    ITcVnImage*  ipSrcImage2,
    ITcVnImage*  ipWeights1,
    ITcVnImage*  ipWeights2,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1925]	First source image (ET_USINT or ET_REAL)
ipSrcImage2	ITcVnImage* [▶ 1925]	Second source image (same type and size as ipSrcImage1)
ipWeights1	ITcVnImage* [▶ 1925]	Weights for ipSrcImage1 (1 channel, ET_REAL)
ipWeights2	ITcVnImage* [▶ 1925]	Weights for ipSrcImage2 (1 channel, ET_REAL)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.20 DivideImageByScalar

Divide each image pixel by a scalar value using saturation arithmetics.

Syntax

Definition:

```
HRESULT DivideImageByScalar(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
```

```
    double       fScalar,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
fScalar	double	Scalar value
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.21 DivideImageByVector

Divides each image pixel by a vector (1 element for each image channel) using saturation arithmetics.

Syntax

Definition:

```
HRESULT DivideImageByVector(
    HRESULT            hrPrev,
    ITcVnImage*        ipSrcImage,
    TcVnVector4_LREAL& aVector,
    ITcVnImage*&       ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
aVector	TcVnVector4_LREAL [▶ 1713]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.22 DivideImages

Element-wise division of two images using saturation arithmetics. (A division by zero equals zero.)

Syntax

Definition:

```
HRESULT DivideImages(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage1,
    ITcVnImage*  ipSrcImage2,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1925]	First source image (dividend)
ipSrcImage2	ITcVnImage* [▶ 1925]	Second source image (divisor)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.23 DivideScalarByImage

Divides a scalar value by each image pixel using saturation arithmetics.

Syntax

Definition:

```
HRESULT DivideScalarByImage(
    HRESULT      hrPrev,
    double       fScalar,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fScalar	double	Scalar value
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.24 DivideVectorByImage

Divides a vector (1 element for each image channel) by each image pixel using saturation arithmetics.

Syntax

Definition:

```
HRESULT DivideVectorByImage(
    HRESULT            hrPrev,
    TcVnVector4_LREAL& aVector,
    ITcVnImage*        ipSrcImage,
    ITcVnImage*&       ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aVector	TcVnVector4_LREAL [▶ 1713]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.25 ElementwiseExp

Computes the natural exponent of each pixel value.

Syntax

Definition:

```
HRESULT ElementwiseExp(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (REAL or LREAL)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (Same type and size as ipSrcImage, an appropriate image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.26 ElementwiseLog

Computes the natural logarithm of each pixel value.

Syntax

Definition:

```
HRESULT ElementwiseLog(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (REAL or LREAL)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (Same type and size as ipSrcImage, an appropriate image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.27 MaxImage

Element-wise maximum of two images.

Syntax

Definition:

```
HRESULT MaxImage(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage1,
```

```
    ITcVnImage*  ipSrcImage2,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1925]	First source image
ipSrcImage2	ITcVnImage* [▶ 1925]	Second source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.28 MaxImage (Exp)

Element-wise maximum of two images.

Syntax

Definition:

```
HRESULT MaxImage(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage1,
    ITcVnImage*  ipSrcImage2,
    ITcVnImage*& ipDestImage,
    ITcVnImage*  ipMask
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1925]	First source image
ipSrcImage2	ITcVnImage* [▶ 1925]	Second source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage* [▶ 1925]	Mask of type USINT

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.29 MaxImageWithScalar

Element-wise maximum of image and scalar value.

Syntax

Definition:

```
HRESULT MaxImageWithScalar(
    HRESULT      hrPrev,
    double       fScalar,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fScalar	double	Scalar value
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.30 MaxImageWithScalar (Exp)

Element-wise maximum of image and scalar value.

Syntax

Definition:

```
HRESULT MaxImageWithScalar(
    HRESULT      hrPrev,
    double       fScalar,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage,
    ITcVnImage*  ipMask
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fScalar	double	Scalar value
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage* [▶ 1925]	Mask of type USINT

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.31 MaxImageWithVector

Element-wise maximum of image and vector (1 element for each image channel).

Syntax

Definition:

```
HRESULT MaxImageWithVector(
    HRESULT            hrPrev,
    TcVnVector4_LREAL& aVector,
    ITcVnImage*        ipSrcImage,
    ITcVnImage*&       ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aVector	TcVnVector4_LREAL [▶ 1713]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.32 MaxImageWithVector (Exp)

Element-wise maximum of image and vector (1 element for each image channel).

Syntax

Definition:

```
HRESULT MaxImageWithVector(
    HRESULT            hrPrev,
    TcVnVector4_LREAL& aVector,
    ITcVnImage*        ipSrcImage,
    ITcVnImage*&       ipDestImage,
    ITcVnImage*        ipMask
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aVector	TcVnVector4_LREAL [▶ 1713]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage* [▶ 1925]	Mask of type USINT

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.33 MinImage

Element-wise minimum of two images.

Syntax

Definition:

```
HRESULT MinImage(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage1,
    ITcVnImage*  ipSrcImage2,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1925]	First source image
ipSrcImage2	ITcVnImage* [▶ 1925]	Second source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.34 MinImage (Exp)

Element-wise minimum of two images.

Syntax

Definition:

```
HRESULT MinImage(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage1,
    ITcVnImage*  ipSrcImage2,
    ITcVnImage*& ipDestImage,
    ITcVnImage*  ipMask
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1925]	First source image
ipSrcImage2	ITcVnImage* [▶ 1925]	Second source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage* [▶ 1925]	Mask of type USINT

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.35 MinImageWithScalar

Element-wise minimum of image and scalar value.

Syntax

Definition:

```
HRESULT MinImageWithScalar(
    HRESULT      hrPrev,
    double       fScalar,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fScalar	double	Scalar value
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.36 MinImageWithScalar (Exp)

Element-wise minimum of image and scalar value.

Syntax

Definition:

```
HRESULT MinImageWithScalar(
    HRESULT      hrPrev,
    double       fScalar,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage,
    ITcVnImage*  ipMask
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fScalar	double	Scalar value
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage* [▶ 1925]	Mask of type USINT

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.37 MinImageWithVector

Element-wise minimum of image and vector (1 element for each image channel).

Syntax

Definition:

```
HRESULT MinImageWithVector(
    HRESULT            hrPrev,
    TcVnVector4_LREAL& aVector,
    ITcVnImage*        ipSrcImage,
    ITcVnImage*&       ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aVector	TcVnVector4_LREAL [▶ 1713]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.38 MinImageWithVector (Exp)

Element-wise minimum of image and vector (1 element for each image channel).

Syntax

Definition:

```
HRESULT MinImageWithVector(
    HRESULT            hrPrev,
    TcVnVector4_LREAL& aVector,
    ITcVnImage*        ipSrcImage,
    ITcVnImage*&       ipDestImage,
    ITcVnImage*        ipMask
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aVector	TcVnVector4_LREAL [▶ 1713]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
ipMask	ITcVnImage* [▶ 1925]	Mask of type USINT

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.39 MultiplyImages

Element-wise multiplication of two images using saturation arithmetics.

Syntax

Definition:

```
HRESULT MultiplyImages(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage1,
    ITcVnImage*  ipSrcImage2,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1925]	First source image
ipSrcImage2	ITcVnImage* [▶ 1925]	Second source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.40 MultiplyImageWithScalar

Multiply each image pixel by a scalar value using saturation arithmetics.

Syntax

Definition:

```
HRESULT MultiplyImageWithScalar(
    HRESULT      hrPrev,
    double       fScalar,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fScalar	double	Scalar value
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.41 MultiplyImageWithVector

Multiply each image pixel by a vector (1 element for each image channel) using saturation arithmetics.

Syntax

Definition:

```
HRESULT MultiplyImageWithVector(
    HRESULT            hrPrev,
    TcVnVector4_LREAL& aVector,
    ITcVnImage*        ipSrcImage,
    ITcVnImage*&       ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aVector	TcVnVector4_LREAL [▶ 1713]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.42 SubtractImageFromScalar

Subtract each image pixel from a scalar value using saturation arithmetics.

Syntax

Definition:

```
HRESULT SubtractImageFromScalar(
    HRESULT      hrPrev,
    double       fScalar,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fScalar	double	Scalar value
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.43 SubtractImageFromVector

Subtract each image pixel from a vector (1 element for each image channel) using saturation arithmetics.

Syntax

Definition:

```
HRESULT SubtractImageFromVector(
    HRESULT            hrPrev,
    TcVnVector4_LREAL& aVector,
    ITcVnImage*        ipSrcImage,
    ITcVnImage*&       ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aVector	TcVnVector4_LREAL [▶ 1713]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.44 SubtractImages

Element-wise subtraction of two images using saturation arithmetics.

Syntax

Definition:

```
HRESULT SubtractImages(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage1,
    ITcVnImage*  ipSrcImage2,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1925]	First source image (minuend)
ipSrcImage2	ITcVnImage* [▶ 1925]	Second source image (subtrahend)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.45 SubtractScalarFromImage

Subtract a scalar value from each image pixel using saturation arithmetics.

Syntax

Definition:

```
HRESULT SubtractScalarFromImage(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    double       fScalar,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
fScalar	double	Scalar value
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.3.46 SubtractVectorFromImage

Subtract a vector (1 element for each image channel) from each image pixel using saturation arithmetics.

Syntax

Definition:

```
HRESULT SubtractVectorFromImage(
    HRESULT            hrPrev,
    ITcVnImage*        ipSrcImage,
    TcVnVector4_LREAL& aVector,
    ITcVnImage*&       ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
aVector	TcVnVector4_LREAL [▶ 1713]&	4-element vector (1 element for each image channel. If the image has less channels, the further vector elements are ignored.)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4 Basic Container Operations

Diese Gruppe enthält Funktionen zur Handhabung von Containern [▶ 163].

Funktionen

Erstellung von Containern

- CopyContainer [▶ 2208]
- CopyContainerElementsConditional_ITcVnContainer [▶ 2208]
- CopyContainerElementsConditional_ITcVnForwardIterator [▶ 2209]
- CreateContainer [▶ 2209]
- CreateContainerFromArray [▶ 2210]
- ExtractContainerRange [▶ 2216]
- ReserveContainerMemory [▶ 2222]

Handhabung von Container-Elementen

- AppendToContainer [▶ 2042]
- CheckIfEmpty [▶ 2205]
- ConvertContainerType [▶ 2207]
- EraseFromContainer [▶ 2211]
- FillContainer(Exp) [▶ 2093]
- GetAt [▶ 2120]
- GetContainer(Exp) [▶ 2217]
- GetNumberOfElements [▶ 2219]
- InsertIntoContainer [▶ 2147]
- ReverseContainer [▶ 2222]
- SetAt [▶ 2176]
- SetContainer [▶ 2223]

Container-Daten in Speicher-Bereich exportieren

- ExportContainer [▶ 2212]
- ExportSubContainer [▶ 2214]
- ExportContainer_String [▶ 2213]
- ExportSubContainer_String [▶ 2215]
- ExportContainerSize [▶ 2213]
- ExportSubContainerSize [▶ 2216]

Handhabung von Iteratoren

- AdvanceIterator [▶ 2205]
- CheckIfIteratorIsAtEnd [▶ 2206]
- GetForwardIterator [▶ 2218]
- GetRandomAccessIterator [▶ 2220]
- IncrementIterator [▶ 2220]
- IteratorDistance [▶ 2221]
- SetIteratorToBegin [▶ 2224]

6.2.3.4.1 AppendToContainer

Dieses Kapitel enthält Funktionen, mit denen ein Element zu einem Container entsprechend dem Datentyp hinzugefügt werden kann.

6.2.3.4.1.1 AppendToContainer (LONG)

Append a single element of type DINT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT         hrPrev,
    LONG            nElement,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nElement	LONG	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.2 AppendToContainer (SHORT)

Append a single element of type INT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT         hrPrev,
    SHORT           nElement,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nElement	SHORT	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.3 AppendToContainer (ITcVnContainer*)

Append a single element to a container or concatenate containers (if ipElement has the same typeId as ipContainer).

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipElement,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipElement	ITcVnContainer* [▶ 1890]	Single element to append to ipContainer or container with several elements to be concatenated
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element(s) will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.4 AppendToContainer (ITcVnForwardIterator*)

Append a single element (represented by an iterator)to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT               hrPrev,
    ITcVnForwardIterator* ipElement,
    ITcVnContainer*       ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipElement	ITcVnForwardIterator* [▶ 1881]	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element(s) will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.5 AppendToContainer (ITcVnImage*)

Append a single element of type ITcVnImage to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT         hrPrev,
    ITcVnImage*     ipElement,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipElement	ITcVnImage* [▶ 1925]	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.6 AppendToContainer (double)

Append a single element of type LREAL to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT         hrPrev,
    double          fElement,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fElement	double	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.7 AppendToContainer (float)

Append a single element of type REAL to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT         hrPrev,
    float           fElement,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fElement	float	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.8 AppendToContainer (char)

Append a single element of type SINT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT         hrPrev,
    char            nElement,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nElement	char	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.9 AppendToContainer (TcVnDMatch)

Append a single element of type TcVnDMatch to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT         hrPrev,
    TcVnDMatch&     stElement,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stElement	TcVnDMatch [▶ 1768]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.10 AppendToContainer (TcVnKeyPoint)

Append a single element of type TcVnKeyPoint to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT         hrPrev,
    TcVnKeyPoint&   stElement,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stElement	TcVnKeyPoint [▶ 1768]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.11 AppendToContainer (TcVnPoint2_DINT)

Append a single element of type TcVnPoint2_DINT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT          hrPrev,
    TcVnPoint2_DINT& aElement,
    ITcVnContainer*  ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint2_DINT [▶ 1711]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.12 AppendToContainer (TcVnPoint2_LREAL)

Append a single element of type TcVnPoint2_LREAL to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT           hrPrev,
    TcVnPoint2_LREAL& aElement,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint2_LREAL [▶ 1711]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.13 AppendToContainer (TcVnPoint2_REAL)

Append a single element of type TcVnPoint2_REAL to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT          hrPrev,
    TcVnPoint2_REAL& aElement,
    ITcVnContainer*  ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint2_REAL [▶ 1711]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.14 AppendToContainer (TcVnPoint3_LREAL)

Append a single element of type TcVnPoint3_LREAL to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT           hrPrev,
    TcVnPoint3_LREAL& aElement,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint3_LREAL [▶ 1711]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.15 AppendToContainer (TcVnPoint3_REAL)

Append a single element of type TcVnPoint3_REAL to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT          hrPrev,
    TcVnPoint3_REAL& aElement,
    ITcVnContainer*  ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint3_REAL [▶ 1711]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.16 AppendToContainer (TcVnRectangle_DINT)

Append a single element of type TcVnRectangle_DINT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT             hrPrev,
    TcVnRectangle_DINT& stElement,
    ITcVnContainer*     ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stElement	TcVnRectangle_DINT [▶ 1781]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.17 AppendToContainer (TcVnVector2_DINT)

Append a single element of type TcVnVector2_DINT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT           hrPrev,
    TcVnVector2_DINT& aElement,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_DINT [▶ 1713]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.18 AppendToContainer (TcVnVector2_INT)

Append a single element of type TcVnVector2_INT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT          hrPrev,
    TcVnVector2_INT& aElement,
    ITcVnContainer*  ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_INT [▶ 1713]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.19 AppendToContainer (TcVnVector2_REAL)

Append a single element of type TcVnVector2_REAL to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT           hrPrev,
    TcVnVector2_REAL& aElement,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_REAL [▶ 1713]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.20 AppendToContainer (TcVnVector2_SINT)

Append a single element of type TcVnVector2_SINT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT           hrPrev,
    TcVnVector2_SINT& aElement,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_SINT [▶ 1713]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.21 AppendToContainer (TcVnVector2_UINT)

Append a single element of type TcVnVector2_UINT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT           hrPrev,
    TcVnVector2_UINT& aElement,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_UINT [▶ 1713]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.22 AppendToContainer (TcVnVector2_USINT)

Append a single element of type TcVnVector2_USINT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT            hrPrev,
    TcVnVector2_USINT& aElement,
    ITcVnContainer*    ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_USINT [▶ 1713]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.23 AppendToContainer (TcVnVector3_INT)

Append a single element of type TcVnVector3_INT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT          hrPrev,
    TcVnVector3_INT& aElement,
    ITcVnContainer*  ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_INT [▶ 1713]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.24 AppendToContainer (TcVnVector3_REAL)

Append a single element of type TcVnVector3_REAL to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT           hrPrev,
    TcVnVector3_REAL& aElement,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_REAL [▶ 1713]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.25 AppendToContainer (TcVnVector3_SINT)

Append a single element of type TcVnVector3_SINT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT           hrPrev,
    TcVnVector3_SINT& aElement,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_SINT [▶ 1713]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.26 AppendToContainer (TcVnVector3_UINT)

Append a single element of type TcVnVector3_UINT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT           hrPrev,
    TcVnVector3_UINT& aElement,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_UINT [▶ 1713]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.27 AppendToContainer (TcVnVector3_USINT)

Append a single element of type TcVnVector3_USINT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT            hrPrev,
    TcVnVector3_USINT& aElement,
    ITcVnContainer*    ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_USINT [▶ 1713]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.28 AppendToContainer (TcVnVector4_DINT)

Append a single element of type TcVnVector4_DINT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT           hrPrev,
    TcVnVector4_DINT& aElement,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_DINT [▶ 1713]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.29 AppendToContainer (TcVnVector4_INT)

Append a single element of type TcVnVector4_INT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT          hrPrev,
    TcVnVector4_INT& aElement,
    ITcVnContainer*  ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_INT [▶ 1713]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.30 AppendToContainer (TcVnVector4_LREAL)

Append a single element of type TcVnVector4_LREAL to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT            hrPrev,
    TcVnVector4_LREAL& aElement,
    ITcVnContainer*    ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_LREAL [▶ 1713]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.31 AppendToContainer (TcVnVector4_SINT)

Append a single element of type TcVnVector4_SINT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT           hrPrev,
    TcVnVector4_SINT& aElement,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_SINT [▶ 1713]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.32 AppendToContainer (TcVnVector4_UINT)

Append a single element of type TcVnVector4_UINT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT           hrPrev,
    TcVnVector4_UINT& aElement,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_UINT [▶ 1713]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.33 AppendToContainer (TcVnVector4_USINT)

Append a single element of type TcVnVector4_USINT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT            hrPrev,
    TcVnVector4_USINT& aElement,
    ITcVnContainer*    ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_USINT [▶ 1713]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.34 AppendToContainer (ULONG)

Append a single element of type UDINT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT         hrPrev,
    ULONG           nElement,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nElement	ULONG	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.35 AppendToContainer (USHORT)

Append a single element of type UINT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT         hrPrev,
    USHORT          nElement,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nElement	USHORT	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.36 AppendToContainer (ULONGLONG)

Append a single element of type ULINT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT         hrPrev,
    ULONGLONG       nElement,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nElement	ULONGLONG	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.37 AppendToContainer (unsigned char)

Append a single element of type USINT to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT         hrPrev,
    unsigned char   nElement,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nElement	unsigned char	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.38 AppendToContainer (TcVnVector4_REAL)

Append a single element of type TcVnVector4_REAL to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT           hrPrev,
    TcVnVector4_REAL& aElement,
    ITcVnContainer*   ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_REAL [▶ 1713]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.1.39 AppendToContainer (TcVnRotatedRectangle)

Append a single element of type TcVnRotatedRectangle to a container.

Syntax

Definition:

```
HRESULT AppendToContainer(
    HRESULT               hrPrev,
    TcVnRotatedRectangle& stElement,
    ITcVnContainer*       ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stElement	TcVnRotatedRectangle [▶ 1782]&	Single element to append to ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container to which the element will be appended

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2 FillContainer

Dieses Kapitel enthält Funktionen, mit denen ein Container entsprechend dem Datentyp mit einem speziellen Wert befüllt werden kann. Diese Funktion ist für Container mit Grundelementen verfügbar.

6.2.3.4.2.1 FillContainer (LONG)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    LONG            nValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
nValue	LONG	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.2 FillContainer (SHORT)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    SHORT           nValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
nValue	SHORT	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.3 FillContainer (ITcVnImage*)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    ITcVnImage*     ipValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
ipValue	ITcVnImage* [▶ 1925]	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.4 FillContainer (double)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    double          fValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
fValue	double	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.5 FillContainer (float)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    float           fValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
fValue	float	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.6 FillContainer (char)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    char            nValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
nValue	char	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.7 FillContainer (TcVnDMatch)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    TcVnDMatch&     stValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
stValue	TcVnDMatch [▶ 1768]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.8 FillContainer (TcVnKeyPoint)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    TcVnKeyPoint&   stValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
stValue	TcVnKeyPoint [▶ 1768]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.9 FillContainer (TcVnPoint2_DINT)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnPoint2_DINT& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnPoint2_DINT [▶ 1711]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.10 FillContainer (TcVnPoint2_LREAL)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnPoint2_LREAL& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnPoint2_LREAL [▶ 1711]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.11 FillContainer (TcVnPoint2_REAL)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnPoint2_REAL& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnPoint2_REAL [▶ 1711]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.12 FillContainer (TcVnPoint3_LREAL)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnPoint3_LREAL& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnPoint3_LREAL [▶ 1711]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.13 FillContainer (TcVnPoint3_REAL)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnPoint3_REAL& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnPoint3_REAL [▶ 1711]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.14 FillContainer (TcVnRectangle_DINT)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT             hrPrev,
    ITcVnContainer*     ipContainer,
    TcVnRectangle_DINT& stValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
stValue	TcVnRectangle_DINT [▶ 1781]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.15 FillContainer (TcVnVector2_DINT)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_DINT& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector2_DINT [▶ 1713]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.16 FillContainer (TcVnVector2_INT)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnVector2_INT& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector2_INT [▶ 1713]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.17 FillContainer (TcVnVector2_REAL)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_REAL& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector2_REAL [▶ 1713]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.18 FillContainer (TcVnVector2_SINT)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_SINT& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector2_SINT [▶ 1713]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.19 FillContainer (TcVnVector2_UINT)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_UINT& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector2_UINT [▶ 1713]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.20 FillContainer (TcVnVector2_USINT)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContainer,
    TcVnVector2_USINT& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector2_USINT [▶ 1713]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.21 FillContainer (TcVnVector3_INT)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnVector3_INT& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector3_INT [▶ 1713]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.22 FillContainer (TcVnVector3_REAL)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector3_REAL& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector3_REAL [▶ 1713]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.23 FillContainer (TcVnVector3_SINT)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector3_SINT& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector3_SINT [▶ 1713]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.24 FillContainer (TcVnVector3_UINT)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector3_UINT& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector3_UINT [▶ 1713]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.25 FillContainer (TcVnVector3_USINT)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContainer,
    TcVnVector3_USINT& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector3_USINT [▶ 1713]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.26 FillContainer (TcVnVector4_DINT)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector4_DINT& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector4_DINT [▶ 1713]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.27 FillContainer (TcVnVector4_INT)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnVector4_INT& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector4_INT [▶ 1713]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.28 FillContainer (TcVnVector4_LREAL)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContainer,
    TcVnVector4_LREAL& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector4_LREAL [▶ 1713]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.29 FillContainer (TcVnVector4_SINT)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector4_SINT& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector4_SINT [▶ 1713]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.30 FillContainer (TcVnVector4_UINT)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector4_UINT& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector4_UINT [▶ 1713]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.31 FillContainer (TcVnVector4_USINT)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContainer,
    TcVnVector4_USINT& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector4_USINT [▶ 1713]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.32 FillContainer (ULONG)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    ULONG           nValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
nValue	ULONG	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.33 FillContainer (USHORT)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    USHORT          nValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
nValue	USHORT	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.34 FillContainer (ULONGLONG)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    ULONGLONG       nValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
nValue	ULONGLONG	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.35 FillContainer (unsigned char)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    unsigned char   nValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
nValue	unsigned char	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.36 FillContainer (TcVnVector4_REAL)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector4_REAL& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector4_REAL [▶ 1713]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.2.37 FillContainer (TcVnRotatedRectangle)

Fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT               hrPrev,
    ITcVnContainer*       ipContainer,
    TcVnRotatedRectangle& stValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
stValue	TcVnRotatedRectangle [▶ 1782]&	Value to set the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3 FillContainerExp

Dieses Kapitel enthält Experten-Funktionen, mit denen ein Container entsprechend dem Datentyp mit einem spezifischen Wert auch nur teilweise befüllt werden kann. Diese Funktion ist für Container mit Grundelementen verfügbar.

6.2.3.4.3.1 FillContainer (LONG)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
```

```
    LONG            nValue,
    ULONGLONG       nStartIdx,
    ULONGLONG       nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
nValue	LONG	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.2 FillContainer (SHORT)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    SHORT           nValue,
    ULONGLONG       nStartIdx,
    ULONGLONG       nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
nValue	SHORT	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.3 FillContainer (ITcVnImage*)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    ITcVnImage*     ipValue,
    ULONGLONG       nStartIdx,
    ULONGLONG       nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
ipValue	ITcVnImage* [▶ 1925]	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.4 FillContainer (double)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT          hrPrev,
    ITcVnContainer* ipContainer,
    double           fValue,
    ULONGLONG        nStartIdx,
    ULONGLONG        nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
fValue	double	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.5 FillContainer (float)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT          hrPrev,
    ITcVnContainer* ipContainer,
    float            fValue,
    ULONGLONG        nStartIdx,
    ULONGLONG        nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
fValue	float	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.6 FillContainer (char)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    char            nValue,
    ULONGLONG       nStartIdx,
    ULONGLONG       nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
nValue	char	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.7 FillContainer (TcVnDMatch)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    TcVnDMatch&     stValue,
    ULONGLONG       nStartIdx,
    ULONGLONG       nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
stValue	TcVnDMatch [▶ 1768]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.8 FillContainer (TcVnKeyPoint)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    TcVnKeyPoint&   stValue,
    ULONGLONG       nStartIdx,
    ULONGLONG       nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
stValue	TcVnKeyPoint [▶ 1768]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.9 FillContainer (TcVnPoint2_DINT)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnPoint2_DINT& aValue,
    ULONGLONG        nStartIdx,
    ULONGLONG        nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnPoint2_DINT [▶ 1711]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.10 FillContainer (TcVnPoint2_LREAL)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnPoint2_LREAL& aValue,
    ULONGLONG         nStartIdx,
    ULONGLONG         nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnPoint2_LREAL [▶ 1711]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.11 FillContainer (TcVnPoint2_REAL)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnPoint2_REAL& aValue,
    ULONGLONG        nStartIdx,
    ULONGLONG        nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnPoint2_REAL [▶ 1711]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.12 FillContainer (TcVnPoint3_LREAL)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnPoint3_LREAL& aValue,
    ULONGLONG         nStartIdx,
    ULONGLONG         nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnPoint3_LREAL [▶ 1711]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.13 FillContainer (TcVnPoint3_REAL)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnPoint3_REAL& aValue,
    ULONGLONG        nStartIdx,
    ULONGLONG        nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnPoint3_REAL [▶ 1711]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.14 FillContainer (TcVnRectangle_DINT)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT             hrPrev,
    ITcVnContainer*     ipContainer,
    TcVnRectangle_DINT& stValue,
    ULONGLONG           nStartIdx,
    ULONGLONG           nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
stValue	TcVnRectangle_DINT [▶ 1781]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.15 FillContainer (TcVnVector2_DINT)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_DINT& aValue,
    ULONGLONG         nStartIdx,
    ULONGLONG         nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector2_DINT [▶ 1713]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.16 FillContainer (TcVnVector2_INT)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_INT& aValue,
    ULONGLONG         nStartIdx,
    ULONGLONG         nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector2_INT [▶ 1713]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.17 FillContainer (TcVnVector2_REAL)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_REAL& aValue,
    ULONGLONG         nStartIdx,
    ULONGLONG         nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector2_REAL [▶ 1713]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.18 FillContainer (TcVnVector2_SINT)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_SINT& aValue,
    ULONGLONG         nStartIdx,
    ULONGLONG         nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector2_SINT [▶ 1713]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.19 FillContainer (TcVnVector2_UINT)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_UINT& aValue,
    ULONGLONG         nStartIdx,
    ULONGLONG         nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector2_UINT [▶ 1713]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.20 FillContainer (TcVnVector2_USINT)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContainer,
    TcVnVector2_USINT& aValue,
    ULONGLONG          nStartIdx,
    ULONGLONG          nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector2_USINT [▶ 1713]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.21 FillContainer (TcVnVector3_INT)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnVector3_INT& aValue,
    ULONGLONG        nStartIdx,
    ULONGLONG        nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector3_INT [▶ 1713]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.22 FillContainer (TcVnVector3_REAL)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector3_REAL& aValue,
    ULONGLONG         nStartIdx,
    ULONGLONG         nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector3_REAL [▶ 1713]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.23 FillContainer (TcVnVector3_SINT)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector3_SINT& aValue,
    ULONGLONG         nStartIdx,
    ULONGLONG         nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector3_SINT [▶ 1713]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.24 FillContainer (TcVnVector3_UINT)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector3_UINT& aValue,
    ULONGLONG         nStartIdx,
    ULONGLONG         nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector3_UINT [▶ 1713]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.25 FillContainer (TcVnVector3_USINT)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContainer,
    TcVnVector3_USINT& aValue,
    ULONGLONG          nStartIdx,
    ULONGLONG          nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector3_USINT [▶ 1713]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.26 FillContainer (TcVnVector4_DINT)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContainer,
    TcVnVector4_DINT& aValue,
    ULONGLONG          nStartIdx,
    ULONGLONG          nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector4_DINT [▶ 1713]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.27 FillContainer (TcVnVector4_INT)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnVector4_INT& aValue,
    ULONGLONG        nStartIdx,
    ULONGLONG        nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector4_INT [▶ 1713]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.28 FillContainer (TcVnVector4_LREAL)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContainer,
    TcVnVector4_LREAL& aValue,
    ULONGLONG          nStartIdx,
    ULONGLONG          nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector4_LREAL [▶ 1713]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.29 FillContainer (TcVnVector4_SINT)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector4_SINT& aValue,
    ULONGLONG         nStartIdx,
    ULONGLONG         nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector4_SINT [▶ 1713]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.30 FillContainer (TcVnVector4_UINT)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector4_UINT& aValue,
    ULONGLONG         nStartIdx,
    ULONGLONG         nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector4_UINT [▶ 1713]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.31 FillContainer (TcVnVector4_USINT)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContainer,
    TcVnVector4_USINT& aValue,
    ULONGLONG          nStartIdx,
    ULONGLONG          nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector4_USINT [▶ 1713]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.32 FillContainer (ULONG)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT          hrPrev,
    ITcVnContainer* ipContainer,
    ULONG            nValue,
    ULONGLONG        nStartIdx,
    ULONGLONG        nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
nValue	ULONG	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.33 FillContainer (USHORT)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT          hrPrev,
    ITcVnContainer* ipContainer,
    USHORT           nValue,
    ULONGLONG        nStartIdx,
    ULONGLONG        nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
nValue	USHORT	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.34 FillContainer (ULONGLONG)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    ULONGLONG       nValue,
    ULONGLONG       nStartIdx,
    ULONGLONG       nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
nValue	ULONGLONG	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.35 FillContainer (unsigned char)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    unsigned char   nValue,
    ULONGLONG       nStartIdx,
    ULONGLONG       nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
nValue	unsigned char	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.36 FillContainer (TcVnVector4_REAL)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector4_REAL& aValue,
    ULONGLONG         nStartIdx,
    ULONGLONG         nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
aValue	TcVnVector4_REAL [▶ 1713]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.3.37 FillContainer (TcVnRotatedRectangle)

Partially fill the container with the specified value. Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT FillContainer(
    HRESULT               hrPrev,
    ITcVnContainer*       ipContainer,
    TcVnRotatedRectangle& stValue,
    ULONGLONG             nStartIdx,
    ULONGLONG             nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
stValue	TcVnRotatedRectangle [▶ 1782]&	Value to set the container elements
nStartIdx	ULONGLONG	Start index
nCount	ULONGLONG	Number of elements to set

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4 GetAt

Die `F_VN_GetAt`-Funktionen sollten verwendet werden, wenn auf einzelne Elemente in einem Container zugegriffen werden muss. Falls alle Elemente eines Containers der Reihe nach verarbeitet werden sollen, bietet sich vielmehr die Verwendung eines Iterators [▶ 2218] an.

6.2.3.4.4.1 GetAt (LONG)

Gets the DINT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    LONG&           nElement,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with DINT elements
nElement	LONG&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.2 GetAt (SHORT)

Gets the INT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    SHORT&          nElement,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with INT elements
nElement	SHORT&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.3 GetAt (ITcVnContainer*)

Gets the ITcVnContainer element at the specified index of the source container. (Alternatively use interface method .GetAt.)

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT          hrPrev,
    ITcVnContainer*  ipSrcContainer,
    ITcVnContainer*& ipDestContainer,
    ULONGLONG        nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1890]	Container with ITcVnContainer elements
ipDestContainer	ITcVnContainer* [▶ 1890]&	Returns the container at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.4 GetAt (ITcVnImage*)

Gets the ITcVnImage element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    ITcVnImage*&    ipElement,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with ITcVnImage elements
ipElement	ITcVnImage* [▶ 1925]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.5 GetAt (double)

Gets the LREAL element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    double&         fElement,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with LREAL elements
fElement	double&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.6 GetAt (float)

Gets the REAL element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
```

```
    float&          fElement,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with REAL elements
fElement	float&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.7 GetAt (char)

Gets the SINT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    char&           nElement,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with SINT elements
nElement	char&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.8 GetAt (TcVnDMatch)

Gets the TcVnDMatch element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    TcVnDMatch&     stElement,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnDMatch elements
stElement	TcVnDMatch [▶ 1768]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.9 GetAt (TcVnKeyPoint)

Gets the TcVnKeyPoint element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    TcVnKeyPoint&   stElement,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnKeyPoint elements
stElement	TcVnKeyPoint [▶ 1768]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.10 GetAt (TcVnPoint2_DINT)

Gets the TcVnPoint2_DINT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnPoint2_DINT& aElement,
    ULONGLONG        nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnPoint2_DINT elements
aElement	TcVnPoint2_DINT [▶ 1711]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.11 GetAt (TcVnPoint2_LREAL)

Gets the TcVnPoint2_LREAL element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnPoint2_LREAL& aElement,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnPoint2_LREAL elements
aElement	TcVnPoint2_LREAL [▶ 1711]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.12 GetAt (TcVnPoint2_REAL)

Gets the TcVnPoint2_REAL element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnPoint2_REAL& aElement,
    ULONGLONG        nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnPoint2_REAL elements
aElement	TcVnPoint2_REAL [▶ 1711]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.13 GetAt (TcVnPoint3_LREAL)

Gets the TcVnPoint3_LREAL element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnPoint3_LREAL& aElement,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnPoint3_LREAL elements
aElement	TcVnPoint3_LREAL [▶ 1711]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.14 GetAt (TcVnPoint3_REAL)

Gets the TcVnPoint3_REAL element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnPoint3_REAL& aElement,
    ULONGLONG        nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnPoint3_REAL elements
aElement	TcVnPoint3_REAL [▶ 1711]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.15 GetAt (TcVnRectangle_DINT)

Gets the TcVnRectangle_DINT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT             hrPrev,
    ITcVnContainer*     ipContainer,
    TcVnRectangle_DINT& stElement,
    ULONGLONG           nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnRectangle_DINT elements
stElement	TcVnRectangle_DINT [▶ 1781]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.16 GetAt (TcVnVector2_DINT)

Gets the TcVnVector2_DINT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_DINT& aElement,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector2_DINT elements
aElement	TcVnVector2_DINT [▶ 1713]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.17 GetAt (TcVnVector2_INT)

Gets the TcVnVector2_INT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_INT&  aElement,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector2_INT elements
aElement	TcVnVector2_INT [▶ 1713]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.18 GetAt (TcVnVector2_REAL)

Gets the TcVnVector2_REAL element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_REAL& aElement,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector2_REAL elements
aElement	TcVnVector2_REAL [▶ 1713]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.19 GetAt (TcVnVector2_SINT)

Gets the TcVnVector2_SINT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_SINT& aElement,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector2_SINT elements
aElement	TcVnVector2_SINT [▶ 1713]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.20 GetAt (TcVnVector2_UINT)

Gets the TcVnVector2_UINT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_UINT& aElement,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector2_UINT elements
aElement	TcVnVector2_UINT [▶ 1713]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.21 GetAt (TcVnVector2_USINT)

Gets the TcVnVector2_USINT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT             hrPrev,
    ITcVnContainer*     ipContainer,
    TcVnVector2_USINT&  aElement,
    ULONGLONG           nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector2_USINT elements
aElement	TcVnVector2_USINT [▶ 1713]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.22 GetAt (TcVnVector3_INT)

Gets the TcVnVector3_INT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnVector3_INT& aElement,
    ULONGLONG        nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector3_INT elements
aElement	TcVnVector3_INT [▶ 1713]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.23 GetAt (TcVnVector3_REAL)

Gets the TcVnVector3_REAL element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector3_REAL& aElement,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector3_REAL elements
aElement	TcVnVector3_REAL [▶ 1713]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.24 GetAt (TcVnVector3_SINT)

Gets the TcVnVector3_SINT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector3_SINT& aElement,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector3_SINT elements
aElement	TcVnVector3_SINT [▶ 1713]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.25 GetAt (TcVnVector3_UINT)

Gets the TcVnVector3_UINT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector3_UINT& aElement,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector3_UINT elements
aElement	TcVnVector3_UINT [▶ 1713]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.26 GetAt (TcVnVector3_USINT)

Gets the TcVnVector3_USINT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT             hrPrev,
    ITcVnContainer*     ipContainer,
    TcVnVector3_USINT&  aElement,
    ULONGLONG           nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector3_USINT elements
aElement	TcVnVector3_USINT [▶ 1713]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.27 GetAt (TcVnVector4_DINT)

Gets the TcVnVector4_DINT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT             hrPrev,
    ITcVnContainer*     ipContainer,
    TcVnVector4_DINT&   aElement,
    ULONGLONG           nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector4_DINT elements
aElement	TcVnVector4_DINT [▶ 1713]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.28 GetAt (TcVnVector4_INT)

Gets the TcVnVector4_INT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnVector4_INT& aElement,
    ULONGLONG        nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector4_INT elements
aElement	TcVnVector4_INT [▶ 1713]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.29 GetAt (TcVnVector4_LREAL)

Gets the TcVnVector4_LREAL element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT             hrPrev,
    ITcVnContainer*     ipContainer,
    TcVnVector4_LREAL&  aElement,
    ULONGLONG           nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector4_LREAL elements
aElement	TcVnVector4_LREAL [▶ 1713]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.30 GetAt (TcVnVector4_SINT)

Gets the TcVnVector4_SINT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector4_SINT& aElement,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector4_SINT elements
aElement	TcVnVector4_SINT [▶ 1713]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.31 GetAt (TcVnVector4_UINT)

Gets the TcVnVector4_UINT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector4_UINT& aElement,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector4_UINT elements
aElement	TcVnVector4_UINT [▶ 1713]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.32 GetAt (TcVnVector4_USINT)

Gets the TcVnVector4_USINT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT             hrPrev,
    ITcVnContainer*     ipContainer,
    TcVnVector4_USINT&  aElement,
    ULONGLONG           nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector4_USINT elements
aElement	TcVnVector4_USINT [▶ 1713]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.33 GetAt (ULONG)

Gets the UDINT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT          hrPrev,
    ITcVnContainer* ipContainer,
    ULONG&           nElement,
    ULONGLONG        nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with UDINT elements
nElement	ULONG&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.34 GetAt (USHORT)

Gets the UINT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT          hrPrev,
    ITcVnContainer* ipContainer,
    USHORT&          nElement,
    ULONGLONG        nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with UINT elements
nElement	USHORT&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.35 GetAt (ULONGLONG)

Gets the ULINT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    ULONGLONG&      nElement,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with ULINT elements
nElement	ULONGLONG&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.36 GetAt (unsigned char)

Gets the USINT element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    unsigned char&  nElement,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with USINT elements
nElement	unsigned char&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.37 GetAt (TcVnVector4_REAL)

Gets the TcVnVector4_REAL element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
```

```
    TcVnVector4_REAL& aElement,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector4_REAL elements
aElement	TcVnVector4_REAL [▶ 1713]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.4.38 GetAt (TcVnRotatedRectangle)

Gets the TcVnRotatedRectangle element at the specified index of the container.

Syntax

Definition:

```
HRESULT GetAt(
    HRESULT               hrPrev,
    ITcVnContainer*       ipContainer,
    TcVnRotatedRectangle& stElement,
    ULONGLONG             nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnRotatedRectangle elements
stElement	TcVnRotatedRectangle [▶ 1782]&	Returns the element at the specified index
nIndex	ULONGLONG	Index of the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5 InsertIntoContainer

Dieses Kapitel enthält Funktionen, mit denen ein Element an einer spezifischen Container-Position entsprechend dem Datentyp eingefügt werden kann.

6.2.3.4.5.1 InsertIntoContainer (LONG)

Insert an element of type DINT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT         hrPrev,
    LONG            nElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nElement	LONG	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.2 InsertIntoContainer (SHORT)

Insert an element of type INT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT         hrPrev,
    SHORT           nElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nElement	SHORT	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.3 InsertIntoContainer (ITcVnContainer*)

Insert an element (or a range of elements) into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipElement	ITcVnContainer* [▶ 1890]	Single element to insert into ipContainer or container with several elements
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.4 InsertIntoContainer (ITcVnForwardIterator*)

Insert an element (represented by an iterator) into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT               hrPrev,
    ITcVnForwardIterator* ipElement,
    ITcVnContainer*       ipContainer,
    ULONGLONG             nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipElement	ITcVnForwardIterator* [▶ 1881]	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.5 InsertIntoContainer (ITcVnImage*)

Insert an element of type ITcVnImage into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT         hrPrev,
    ITcVnImage*     ipElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipElement	ITcVnImage* [▶ 1925]	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.6 InsertIntoContainer (double)

Insert an element of type LREAL into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT         hrPrev,
    double          fElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fElement	double	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.7 InsertIntoContainer (float)

Insert an element of type REAL into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT         hrPrev,
    float           fElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fElement	float	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.8 InsertIntoContainer (char)

Insert an element of type SINT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT         hrPrev,
    char            nElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nElement	char	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.9 InsertIntoContainer (TcVnDMatch)

Insert an element of type TcVnDMatch into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT         hrPrev,
    TcVnDMatch&     stElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stElement	TcVnDMatch [▶ 1768]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.10 InsertIntoContainer (TcVnKeyPoint)

Insert an element of type TcVnKeyPoint into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT         hrPrev,
    TcVnKeyPoint&   stElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stElement	TcVnKeyPoint [▶ 1768]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.11 InsertIntoContainer (TcVnPoint2_DINT)

Insert an element of type TcVnPoint2_DINT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT          hrPrev,
    TcVnPoint2_DINT& aElement,
    ITcVnContainer*  ipContainer,
    ULONGLONG        nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint2_DINT [▶ 1711]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.12 InsertIntoContainer (TcVnPoint2_LREAL)

Insert an element of type TcVnPoint2_LREAL into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT           hrPrev,
    TcVnPoint2_LREAL& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint2_LREAL [▶ 1711]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.13 InsertIntoContainer (TcVnPoint2_REAL)

Insert an element of type TcVnPoint2_REAL into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT          hrPrev,
    TcVnPoint2_REAL& aElement,
    ITcVnContainer*  ipContainer,
    ULONGLONG        nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint2_REAL [▶ 1711]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.14 InsertIntoContainer (TcVnPoint3_LREAL)

Insert an element of type TcVnPoint3_LREAL into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT           hrPrev,
    TcVnPoint3_LREAL& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint3_LREAL [▶ 1711]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.15 InsertIntoContainer (TcVnPoint3_REAL)

Insert an element of type TcVnPoint3_REAL into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT          hrPrev,
    TcVnPoint3_REAL& aElement,
    ITcVnContainer*  ipContainer,
    ULONGLONG        nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint3_REAL [▶ 1711]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.16 InsertIntoContainer (TcVnRectangle_DINT)

Insert an element of type TcVnRectangle_DINT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT             hrPrev,
    TcVnRectangle_DINT& stElement,
    ITcVnContainer*     ipContainer,
    ULONGLONG           nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stElement	TcVnRectangle_DINT [▶ 1781]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.17 InsertIntoContainer (TcVnVector2_DINT)

Insert an element of type TcVnVector2_DINT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT           hrPrev,
    TcVnVector2_DINT& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_DINT [▶ 1713]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.18 InsertIntoContainer (TcVnVector2_INT)

Insert an element of type TcVnVector2_INT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT          hrPrev,
    TcVnVector2_INT& aElement,
    ITcVnContainer*  ipContainer,
    ULONGLONG        nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_INT [▶ 1713]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.19 InsertIntoContainer (TcVnVector2_REAL)

Insert an element of type TcVnVector2_REAL into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT           hrPrev,
    TcVnVector2_REAL& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_REAL [▶ 1713]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.20 InsertIntoContainer (TcVnVector2_SINT)

Insert an element of type TcVnVector2_SINT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT           hrPrev,
    TcVnVector2_SINT& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_SINT [▶ 1713]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.21 InsertIntoContainer (TcVnVector2_UINT)

Insert an element of type TcVnVector2_UINT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT           hrPrev,
    TcVnVector2_UINT& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_UINT [▶ 1713]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.22 InsertIntoContainer (TcVnVector2_USINT)

Insert an element of type TcVnVector2_USINT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT            hrPrev,
    TcVnVector2_USINT& aElement,
    ITcVnContainer*    ipContainer,
    ULONGLONG          nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_USINT [▶ 1713]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.23 InsertIntoContainer (TcVnVector3_INT)

Insert an element of type TcVnVector3_INT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT          hrPrev,
    TcVnVector3_INT& aElement,
    ITcVnContainer*  ipContainer,
    ULONGLONG        nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_INT [▶ 1713]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.24 InsertIntoContainer (TcVnVector3_REAL)

Insert an element of type TcVnVector3_REAL into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT           hrPrev,
    TcVnVector3_REAL& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_REAL [▶ 1713]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.25 InsertIntoContainer (TcVnVector3_SINT)

Insert an element of type TcVnVector3_SINT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT           hrPrev,
    TcVnVector3_SINT& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_SINT [▶ 1713]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.26 InsertIntoContainer (TcVnVector3_UINT)

Insert an element of type TcVnVector3_UINT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT           hrPrev,
    TcVnVector3_UINT& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_UINT [▶ 1713]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.27 InsertIntoContainer (TcVnVector3_USINT)

Insert an element of type TcVnVector3_USINT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT            hrPrev,
    TcVnVector3_USINT& aElement,
    ITcVnContainer*    ipContainer,
    ULONGLONG          nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_USINT [▶ 1713]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.28 InsertIntoContainer (TcVnVector4_DINT)

Insert an element of type TcVnVector4_DINT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT           hrPrev,
    TcVnVector4_DINT& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_DINT [▶ 1713]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.29 InsertIntoContainer (TcVnVector4_INT)

Insert an element of type TcVnVector4_INT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT          hrPrev,
    TcVnVector4_INT& aElement,
    ITcVnContainer*  ipContainer,
    ULONGLONG        nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_INT [▶ 1713]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.30 InsertIntoContainer (TcVnVector4_LREAL)

Insert an element of type TcVnVector4_LREAL into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT            hrPrev,
    TcVnVector4_LREAL& aElement,
    ITcVnContainer*    ipContainer,
    ULONGLONG          nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_LREAL [▶ 1713]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.31 InsertIntoContainer (TcVnVector4_SINT)

Insert an element of type TcVnVector4_SINT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT           hrPrev,
    TcVnVector4_SINT& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_SINT [▶ 1713]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.32 InsertIntoContainer (TcVnVector4_UINT)

Insert an element of type TcVnVector4_UINT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT           hrPrev,
    TcVnVector4_UINT& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_UINT [▶ 1713]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.33 InsertIntoContainer (TcVnVector4_USINT)

Insert an element of type TcVnVector4_USINT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT            hrPrev,
    TcVnVector4_USINT& aElement,
    ITcVnContainer*    ipContainer,
    ULONGLONG          nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_USINT [▶ 1713]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.34 InsertIntoContainer (ULONG)

Insert an element of type UDINT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT         hrPrev,
    ULONG           nElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nElement	ULONG	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.35 InsertIntoContainer (USHORT)

Insert an element of type UINT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT         hrPrev,
    USHORT          nElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nElement	USHORT	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.36 InsertIntoContainer (ULONGLONG)

Insert an element of type ULINT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT         hrPrev,
    ULONGLONG       nElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nElement	ULONGLONG	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.37 InsertIntoContainer (unsigned char)

Insert an element of type USINT into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT         hrPrev,
    unsigned char   nElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nElement	unsigned char	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.38 InsertIntoContainer (TcVnVector4_REAL)

Insert an element of type TcVnVector4_REAL into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT           hrPrev,
    TcVnVector4_REAL& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_REAL [▶ 1713]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.5.39 InsertIntoContainer (TcVnRotatedRectangle)

Insert an element of type TcVnRotatedRectangle into a container before the specified position.

Syntax

Definition:

```
HRESULT InsertIntoContainer(
    HRESULT               hrPrev,
    TcVnRotatedRectangle& stElement,
    ITcVnContainer*       ipContainer,
    ULONGLONG             nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stElement	TcVnRotatedRectangle [▶ 1782]&	Single element to insert into ipContainer
ipContainer	ITcVnContainer* [▶ 1890]	Container in which to insert the element
nIndex	ULONGLONG	Position, before which the element will be inserted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6 SetAt

Die `F_VN_SetAt`-Funktionen sollten verwendet werden, wenn auf einzelne Elemente in einem Container zugegriffen werden muss. Falls alle Elemente eines Containers der Reihe nach bearbeitet werden sollen, bietet sich vielmehr die Verwendung eines Iterators [▶ 2218] an.

6.2.3.4.6.1 SetAt (LONG)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT         hrPrev,
    LONG            nElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nElement	LONG	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with DINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.2 SetAt (SHORT)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT         hrPrev,
    SHORT           nElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nElement	SHORT	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with INT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.3 SetAt (ITcVnContainer*)

Sets the element at the specified index of the container. (Alternatively use interface method .SetAt.)

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT         hrPrev,
    ITcVnContainer* ipElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipElement	ITcVnContainer* [▶ 1890]	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with ITcVnContainer elements, in which the element at position nIndex is replaced by ipElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.4 SetAt (ITcVnImage*)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT         hrPrev,
    ITcVnImage*     ipElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipElement	ITcVnImage* [▶ 1925]	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with ITcVnImage elements, in which the element at position nIndex is replaced by ipElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.5 SetAt (double)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT         hrPrev,
    double          fElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fElement	double	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with LREAL elements, in which the element at position nIndex is replaced by fElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.6 SetAt (float)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT         hrPrev,
    float           fElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fElement	float	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with REAL elements, in which the element at position nIndex is replaced by fElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.7 SetAt (char)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT         hrPrev,
    char            nElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nElement	char	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with SINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.8 SetAt (TcVnDMatch)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT          hrPrev,
    TcVnDMatch&      stElement,
    ITcVnContainer* ipContainer,
    ULONGLONG        nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stElement	TcVnDMatch [▶ 1768]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnDMatch elements, in which the element at position nIndex is replaced by stElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.9 SetAt (TcVnKeyPoint)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT          hrPrev,
    TcVnKeyPoint&    stElement,
    ITcVnContainer* ipContainer,
    ULONGLONG        nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stElement	TcVnKeyPoint [▶ 1768]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnKeyPoint elements, in which the element at position nIndex is replaced by stElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.10 SetAt (TcVnPoint2_DINT)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT          hrPrev,
    TcVnPoint2_DINT& aElement,
    ITcVnContainer*  ipContainer,
    ULONGLONG        nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint2_DINT [▶ 1711]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnPoint2_DINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.11 SetAt (TcVnPoint2_LREAL)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT           hrPrev,
    TcVnPoint2_LREAL& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint2_LREAL [▶ 1711]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnPoint2_LREAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.12 SetAt (TcVnPoint2_REAL)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT          hrPrev,
    TcVnPoint2_REAL& aElement,
    ITcVnContainer*  ipContainer,
    ULONGLONG        nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint2_REAL [▶ 1711]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnPoint2_REAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.13 SetAt (TcVnPoint3_LREAL)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT           hrPrev,
    TcVnPoint3_LREAL& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint3_LREAL [▶ 1711]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnPoint3_LREAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.14 SetAt (TcVnPoint3_REAL)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT          hrPrev,
    TcVnPoint3_REAL& aElement,
    ITcVnContainer*  ipContainer,
    ULONGLONG        nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnPoint3_REAL [▶ 1711]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnPoint3_REAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.15 SetAt (TcVnRectangle_DINT)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT             hrPrev,
    TcVnRectangle_DINT& stElement,
    ITcVnContainer*     ipContainer,
    ULONGLONG           nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stElement	TcVnRectangle_DINT [▶ 1781]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnRectangle_DINT elements, in which the element at position nIndex is replaced by stElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.16 SetAt (TcVnVector2_DINT)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT           hrPrev,
    TcVnVector2_DINT& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_DINT [▶ 1713]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector2_DINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.17 SetAt (TcVnVector2_INT)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT          hrPrev,
    TcVnVector2_INT& aElement,
    ITcVnContainer*  ipContainer,
    ULONGLONG        nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_INT [▶ 1713]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector2_INT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.18 SetAt (TcVnVector2_REAL)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT           hrPrev,
    TcVnVector2_REAL& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_REAL [▶ 1713]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector2_REAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.19 SetAt (TcVnVector2_SINT)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT           hrPrev,
    TcVnVector2_SINT& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_SINT [▶ 1713]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector2_SINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.20 SetAt (TcVnVector2_UINT)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT           hrPrev,
    TcVnVector2_UINT& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_UINT [▶ 1713]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector2_UINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.21 SetAt (TcVnVector2_USINT)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT            hrPrev,
    TcVnVector2_USINT& aElement,
    ITcVnContainer*    ipContainer,
    ULONGLONG          nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector2_USINT [▶ 1713]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector2_USINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.22 SetAt (TcVnVector3_INT)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT          hrPrev,
    TcVnVector3_INT& aElement,
    ITcVnContainer*  ipContainer,
    ULONGLONG        nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_INT [▶ 1713]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector3_INT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.23 SetAt (TcVnVector3_REAL)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT           hrPrev,
    TcVnVector3_REAL& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_REAL [▶ 1713]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector3_REAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.24 SetAt (TcVnVector3_SINT)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT           hrPrev,
    TcVnVector3_SINT& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_SINT [▶ 1713]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector3_SINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.25 SetAt (TcVnVector3_UINT)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT           hrPrev,
    TcVnVector3_UINT& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_UINT [▶ 1713]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector3_UINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.26 SetAt (TcVnVector3_USINT)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT            hrPrev,
    TcVnVector3_USINT& aElement,
    ITcVnContainer*    ipContainer,
    ULONGLONG          nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector3_USINT [▶ 1713]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector3_USINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.27 SetAt (TcVnVector4_DINT)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT           hrPrev,
    TcVnVector4_DINT& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_DINT [▶ 1713]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector4_DINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.28 SetAt (TcVnVector4_INT)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT          hrPrev,
    TcVnVector4_INT& aElement,
    ITcVnContainer*  ipContainer,
    ULONGLONG        nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_INT [▶ 1713]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector4_INT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.29 SetAt (TcVnVector4_LREAL)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT            hrPrev,
    TcVnVector4_LREAL& aElement,
    ITcVnContainer*    ipContainer,
    ULONGLONG          nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_LREAL [▶ 1713]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector4_LREAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.30 SetAt (TcVnVector4_SINT)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT           hrPrev,
    TcVnVector4_SINT& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_SINT [▶ 1713]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector4_SINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.31 SetAt (TcVnVector4_UINT)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT           hrPrev,
    TcVnVector4_UINT& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_UINT [▶ 1713]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector4_UINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.32 SetAt (TcVnVector4_USINT)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT            hrPrev,
    TcVnVector4_USINT& aElement,
    ITcVnContainer*    ipContainer,
    ULONGLONG          nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_USINT [▶ 1713]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector4_USINT elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.33 SetAt (ULONG)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT         hrPrev,
    ULONG           nElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nElement	ULONG	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with UDINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.34 SetAt (USHORT)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT         hrPrev,
    USHORT          nElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nElement	USHORT	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with UINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.35 SetAt (ULONGLONG)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT         hrPrev,
    ULONGLONG       nElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nElement	ULONGLONG	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with ULINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.36 SetAt (unsigned char)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT         hrPrev,
    unsigned char   nElement,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nElement	unsigned char	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with USINT elements, in which the element at position nIndex is replaced by nElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.37 SetAt (TcVnVector4_REAL)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT           hrPrev,
    TcVnVector4_REAL& aElement,
    ITcVnContainer*   ipContainer,
    ULONGLONG         nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aElement	TcVnVector4_REAL [▶ 1713]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnVector4_REAL elements, in which the element at position nIndex is replaced by aElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.6.38 SetAt (TcVnRotatedRectangle)

Sets the element at the specified index of the container.

Syntax

Definition:

```
HRESULT SetAt(
    HRESULT               hrPrev,
    TcVnRotatedRectangle& stElement,
    ITcVnContainer*       ipContainer,
    ULONGLONG             nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stElement	TcVnRotatedRectangle [▶ 1782]&	Element to set at the specified container position
ipContainer	ITcVnContainer* [▶ 1890]	Container with TcVnRotatedRectangle elements, in which the element at position nIndex is replaced by stElement
nIndex	ULONGLONG	Index

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.7 AdvanceIterator

Advance an iterator by the specified offset.

Syntax

Definition:

```
HRESULT AdvanceIterator(
    HRESULT               hrPrev,
    ITcVnForwardIterator* ipIterator,
    LONGLONG              nOffset
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipIterator	ITcVnForwardIterator* [▶ 1881]	Iterator to be advanced
nOffset	LONGLONG	Offset, by which ipIterator is advanced

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.8 CheckIfEmpty

Checks if the container is empty. (Alternatively use interface method .CheckIfEmpty.)

Syntax

Definition:

```
HRESULT CheckIfEmpty(
    HRESULT          hrPrev,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.9 CheckIfIteratorIsAtEnd

Checks if the iterator points to the past-the-end element. (Alternatively use interface method .CheckIfEnd.)

Syntax

Definition:

```
HRESULT CheckIfIteratorIsAtEnd(
    HRESULT               hrPrev,
    ITcVnForwardIterator* ipIterator
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipIterator	ITcVnForwardIterator* [▶ 1881]	Iterator

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.10 ConvertContainerType

Converts a container to another type (Struct element types are not supported).

Syntax

Definition:

```
HRESULT ConvertContainerType(
    HRESULT          hrPrev,
    ITcVnContainer*  ipSrcContainer,
    ITcVnContainer*& ipDestContainer,
    GUID             nDestTypeGuid
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1890]	Source container
ipDestContainer	ITcVnContainer* [▶ 1890]&	Returns the converted container
nDestTypeGuid	GUID	Specifies the destination container type for the conversion

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.11 CopyContainer

Copys a container.

Syntax

Definition:

```
HRESULT CopyContainer(
    HRESULT          hrPrev,
    ITcVnContainer*  ipSrcContainer,
    ITcVnContainer*& ipDestContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1890]	Source container
ipDestContainer	ITcVnContainer* [▶ 1890]&	Destination container (same type ID as ipSrcContainer; Non-zero interface pointers are reused.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.12 CopyContainerElementsConditional (ITcVnContainer*)

Copys container elements to a new container, depending on a custom condition.

Syntax

Definition:

```
HRESULT CopyContainerElementsConditional(
    HRESULT                                    hrPrev,
    ITcVnContainer*                            ipSrcContainer,
    ITcVnContainer*&                           ipDestContainer,
    ITcVnCustomElementCondition_ITcVnContainer* ipConditionFB
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1890]	Source container
ipDestContainer	ITcVnContainer* [▶ 1890]&	Returns a container with elements, that match the condition
ipConditionFB	ITcVnCustomElementCondition_ITcVnContainer* [▶ 1785]	Custom condition

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.13 CopyContainerElementsConditional (ITcVnForwardIterator*)

Copys container elements to a new container, depending on a custom condition.

Syntax

Definition:

```
HRESULT CopyContainerElementsConditional(
    HRESULT                                          hrPrev,
    ITcVnContainer*                                  ipSrcContainer,
    ITcVnContainer*&                                 ipDestContainer,
    ITcVnCustomElementCondition_ITcVnForwardIterator* ipConditionFB
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1890]	Source container
ipDestContainer	ITcVnContainer* [▶ 1890]&	Returns a container with elements, that match the condition
ipConditionFB	ITcVnCustomElementCondition_ITcVnForwardIterator* [▶ 1786]	Custom condition

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.14 CreateContainer

Creates a container with type GUID nTypeGuid and return its container interface.

Syntax

Definition:

```
HRESULT CreateContainer(
    HRESULT          hrPrev,
    ITcVnContainer*& ipContainer,
    GUID             nTypeGuid,
    ULONGLONG        nElementNum
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]&	Returns the created container
nTypeGuid	GUID	Type GUID of the container to be created
nElementNum	ULONGLONG	Number of elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.15 CreateContainerFromArray

Creates a container with type GUID nTypeGuid, initialize it with provided data and return its container interface. Only supported for basic container types, i.e. not for containers of containers.

Syntax

Definition:

```
HRESULT CreateContainerFromArray(
    HRESULT          hrPrev,
    PVOID            pData,
    ITcVnContainer*& ipContainer,
    GUID             nTypeGuid,
    ULONGLONG        nElementNum
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
pData	PVOID	Pointer to the data array. Make sure that the array contains at least nElementNum elements and that the array element type matches the container element type.
ipContainer	ITcVnContainer* [▶ 1890]&	Returns the created container
nTypeGuid	GUID	Type GUID of the container to be created
nElementNum	ULONGLONG	Number of elements to copy from pData

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.16 EraseFromContainer

Erase elements from a container.

Syntax

Definition:

```
HRESULT EraseFromContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    ULONGLONG       nStartIdx,
    ULONGLONG       nCount
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container from which to erase elements
nStartIdx	ULONGLONG	Index of the first element to erase
nCount	ULONGLONG	Number of elements to erase

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.17 ExportContainer

Export the container elements into a buffer (e.g. an array). Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT ExportContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    PVOID           pBuffer,
    ULONGLONG       nBufferSize
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
pBuffer	PVOID	Buffer to store the container elements (Make sure to allocate enough memory! The required size in bytes can be determined using the function F_VN_ExportContainerSize)
nBufferSize	ULONGLONG	Size of the buffer memory in bytes

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.18 ExportContainer_String

Export the container elements into a string. Only possible for containers of type ContainerType_String_SINT.

Syntax

Definition:

```
HRESULT ExportContainer_String(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    PCHAR           sText,
    ULONGLONG       nMaxLength
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container of type ContainerType_String_SINT
sText	PCHAR	Make sure to choose a sufficient STRING size! The required size can be determined using the function VnExportContainerSize.
nMaxLength	ULONGLONG	Maximum string length to export (including 0 termination). If the container content is longer, the string is cut off at nMaxLength - 1 and 0 termination is appended. In this case, S_FALSE is returned.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.19 ExportContainerSize

Determine the required buffer size in bytes to store all container elements (number_of_Elements * size_per_Element). Only possible for containers with basic elements.

Syntax

Definition:

```
HRESULT ExportContainerSize(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    ULONGLONG&      nBufferSize
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
nBufferSize	ULONGLONG&	Output parameter containing the required buffer size

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.20 ExportSubContainer

Export the container elements of a sub-container into a buffer (e.g. an array). Only possible for 2-dimensional containers with basic elements.

Syntax

Definition:

```
HRESULT ExportSubContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex,
    PVOID           pBuffer,
    ULONGLONG       nBufferSize
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
nIndex	ULONGLONG	Index of the requested element
pBuffer	PVOID	Buffer to store the container elements (Make sure to allocate enough memory! The required size in bytes can be determined using the function F_VN_ExportSubContainerSize)
nBufferSize	ULONGLONG	Size of the buffer memory in bytes

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.21 ExportSubContainer_String

Export the container elements of a sub-container into a string. Only possible for 2-dimensional containers of type ContainerType_Vector_String_SINT.

Syntax

Definition:

```
HRESULT ExportSubContainer_String(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex,
    PCHAR           sText,
    ULONGLONG       nMaxLength
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container of type ContainerType_Vector_String_SINT
nIndex	ULONGLONG	Index of the requested element
sText	PCHAR	Make sure to choose a sufficient STRING size! The required size can be determined using the function VnExportSubContainerSize.
nMaxLength	ULONGLONG	Maximum string length to export (including 0 termination). If the container content is longer, the string is cut off at nMaxLength - 1 and 0 termination is appended. In this case, S_FALSE is returned.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.22 ExportSubContainerSize

Determine the required buffer size in bytes to store all container elements of a sub-container (number_of_Elements * size_per_Element). Only possible for 2-dimensional containers with basic elements.

Syntax

Definition:

```
HRESULT ExportSubContainerSize(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    ULONGLONG       nIndex,
    ULONGLONG&      nBufferSize
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container with basic elements
nIndex	ULONGLONG	Index of the requested element
nBufferSize	ULONGLONG&	Output parameter containing the required buffer size

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.23 ExtractContainerRange

Copy the specified range of the source container into the destination container. (If the destination container already contains elements, they will be removed.)

Syntax

Definition:

```
HRESULT ExtractContainerRange(
    HRESULT          hrPrev,
    ITcVnContainer*  ipSrcContainer,
    ITcVnContainer*& ipDestContainer,
```

```
    ULONGLONG        nFirstIdx,
    ULONGLONG        nLastIdx
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1890]	Source container
ipDestContainer	ITcVnContainer* [▶ 1890]&	Destination container
nFirstIdx	ULONGLONG	Index of the first element to copy
nLastIdx	ULONGLONG	Index of the last element to copy

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.24 GetContainer

Gets a pointer to the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). (Alternatively use interface method .GetContainer.)

Syntax

Definition:

```
HRESULT GetContainer(
    HRESULT               hrPrev,
    ITcVnForwardIterator* ipIterator,
    ITcVnContainer*&      ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipIterator	ITcVnForwardIterator* [▶ 1881]	Iterator
ipContainer	ITcVnContainer* [▶ 1890]&	Returns the container interface

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.25 GetContainer (Exp)

Gets a pointer to the current element converted into an ITcVnContainer interface and increment its reference counter (only possible for container types). Additionally, an offset to increment or decrement the iterator afterwards can be provided.

Syntax

Definition:

```
HRESULT GetContainer(
    HRESULT               hrPrev,
    ITcVnForwardIterator* ipIterator,
    ITcVnContainer*&      ipContainer,
    LONG                  nOffset
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipIterator	ITcVnForwardIterator* [▶ 1881]	Iterator
ipContainer	ITcVnContainer* [▶ 1890]&	Returns the container interface
nOffset	LONG	Offset to increment (>0) or decrement (<0) the iterator afterwards

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.26 GetForwardIterator

Gets a forward iterator for the container. (Alternatively use interface method .GetForwardIterator.)

Syntax

Definition:

```
HRESULT GetForwardIterator(
    HRESULT                hrPrev,
    ITcVnContainer*        ipContainer,
    ITcVnForwardIterator*& ipIterator
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container
ipIterator	ITcVnForwardIterator* [▶ 1881]&	Returns the iterator interface

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.27 GetNumberOfElements

Gets the number of elements in the container. (Alternatively use interface method .getElementNum.)

Syntax

Definition:

```
HRESULT GetNumberOfElements(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    ULONGLONG&      nNumberOfElements
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container
nNumberOfElements	ULONGLONG&	Returns the number of elements in ipContainer

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.28 GetRandomAccessIterator

Gets a random access iterator for the container. (Alternatively use interface method .GetRandomAccessIterator.)

Syntax

Definition:

```
HRESULT GetRandomAccessIterator(
    HRESULT                     hrPrev,
    ITcVnContainer*             ipContainer,
    ITcVnRandomAccessIterator*& ipIterator
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container
ipIterator	ITcVnRandomAccessIterator* [▶ 1887]&	Returns the iterator interface

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.29 IncrementIterator

Increment the iterator. (Alternatively use interface method .Increment.)

Syntax

Definition:

```
HRESULT IncrementIterator(
    HRESULT               hrPrev,
    ITcVnForwardIterator* ipIterator
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipIterator	ITcVnForwardIterator* [▶ 1881]	Iterator

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.30 IteratorDistance

Computes the distance between 2 iterators.

Syntax

Definition:

```
HRESULT IteratorDistance(
    HRESULT               hrPrev,
    ITcVnForwardIterator* ipIteratorFirst,
    ITcVnForwardIterator* ipIteratorLast,
    LONGLONG&             nDistance
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipIteratorFirst	ITcVnForwardIterator* [▶ 1881]	First iterator
ipIteratorLast	ITcVnForwardIterator* [▶ 1881]	Last iterator
nDistance	LONGLONG&	Returns the computed distance

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.31 ReserveContainerMemory

Reserve container memory (call with maximum required number of elements before manually appending elements for better performance)

Syntax

Definition:

```
HRESULT ReserveContainerMemory(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    ULONGLONG       nElements
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container for which to reserve the memory
nElements	ULONGLONG	Number of elements for which the container should reserve memory

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.32 ReverseContainer

Reverse the container elements.

Syntax

Definition:

```
HRESULT ReverseContainer(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Container to be reversed

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.33 SetContainer

Sets the current element using an ITcVnContainer interface (only possible for container types). (Alternatively use interface method .SetContainer.)

Syntax

Definition:

```
HRESULT SetContainer(
    HRESULT               hrPrev,
    ITcVnForwardIterator* ipIterator,
    ITcVnContainer*       ipContainer
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipIterator	ITcVnForwardIterator* [▶ 1881]	Iterator
ipContainer	ITcVnContainer* [▶ 1890]	Container interface of which the content is to be assigned to the current element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.4.34 SetIteratorToBegin

Sets the iterator to the first element of the container. (Alternatively use interface method .SetToBegin.)

Syntax

Definition:

```
HRESULT SetIteratorToBegin(
    HRESULT               hrPrev,
    ITcVnForwardIterator* ipIterator
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipIterator	ITcVnForwardIterator* [▶ 1881]	Iterator

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5 Basic Image Operations

Die Gruppe enthält Funktionen zur Handhabung von Bildern [▶ 159].

Funktionen

Erstellung von Bildern

- ConvertElementType [▶ 2226]
- CreateAssociatedImage [▶ 2231]
- CreateEmptyImage [▶ 2231]
- CreateImage [▶ 2232]
- CreateImageAndSetPixels [▶ 2233]

- CreateImageFromArray [▶ 2234]
- CreateNDimensionalImage [▶ 2253]
- CreateNDimensionalImageAndSetPixels [▶ 2254]
- CreateNDimensionalImageFromArray [▶ 2255]

Bild-Informationen

- GetImageHeight [▶ 2241]
- GetImageInfo [▶ 2242]
- GetImageWidth [▶ 2242]
- GetLayoutInfo [▶ 2255]
- GetPixelFormat [▶ 2244]

Kopieren von Bildern

- CopyImage [▶ 2227]
- CopyImageRegion [▶ 2228]
- CopyImageRegionToRegion [▶ 2229]

Pixel-Manipulation

- GetPixel [▶ 2243]
- SetPixel [▶ 2247]
- SetPixels [▶ 2248]

Region of Interest

- GetRoi [▶ 2244]
- ResetRoi [▶ 2246]
- SetRoi [▶ 2249]
- SetRoi_TcVnRectangle_DINT [▶ 2250]
- SetRoi_TcVnRectangle_UDINT [▶ 2250]

Bildinhalt in Speicher-Bereich exportieren

- ExportImage [▶ 2235]
- ExportImageAsBmp [▶ 2236]
- ExportImageSize [▶ 2238]
- ExportImageAsBmpSize [▶ 2237]

Bilder fusionieren

- FuseImages [▶ 2239]
- FuseImagesArray [▶ 2240]

Handhabung von Bilder-Kanälen

- CombineImageChannels [▶ 2226]
- GetImageChannel [▶ 2240]
- MixImageChannels [▶ 2245]
- SetImageChannel [▶ 2247]
- SplitImageChannels [▶ 2251]

Anzeigen von Bildern

- CopyIntoDisplayableImage [▶ 2230]
- TransformIntoDisplayableImage [▶ 2252]

Sonstiges

- PadImageBorder [▶ 2256]
- ReinterpretUnsupportedImage [▶ 2257]

6.2.3.5.1 CombineImageChannels

Combines several single-channel images into one multi-channel image.

Syntax

Definition:

```
HRESULT CombineImageChannels(
    HRESULT      hrPrev,
    PVOID        pSrcImages,
    ITcVnImage*& ipDestImage,
    USHORT       nSrcArraySize
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
pSrcImages	PVOID	Pointer to an array of single-channel source images
ipDestImage	ITcVnImage* [▶ 1925]&	Multi-channel destination image (An appropriate destination image will be created if required.)
nSrcArraySize	USHORT	pSrcImages array size

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.2 ConvertElementType

Converts an image to another element type.

Syntax

Definition:

```
HRESULT ConvertElementType(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage,
```

```
    ITcVnImage*&     ipDestImage,
    ETcVnElementType eElementType,
    double           fScaleFactor = 1,
    double           fDelta = 0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (An appropriate destination image will be created if required.)
eElementType	ETcVnElementType [▶ 1738]		Desired element type of the destination image
fScaleFactor	double	1	Scale factor for the pixel values
fDelta	double	0	Value that is added to the scaled pixel values

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.3 CopyImage

Creates a deep copy of an image.

Syntax

Definition:

```
HRESULT CopyImage(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.4 CopyImageRegion

Deep copy the specified region of interest into a new image.

Syntax

Definition:

```
HRESULT CopyImageRegion(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ULONG        nX,
    ULONG        nY,
    ULONG        nWidth,
    ULONG        nHeight,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
nX	ULONG	Left boundary (inclusive 0-based index)
nY	ULONG	Upper boundary (inclusive 0-based index)
nWidth	ULONG	ROI width
nHeight	ULONG	ROI height
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.5 CopyImageRegionToRegion

Copy an image region into another image region.

Syntax

Definition:

```
HRESULT CopyImageRegionToRegion(
    HRESULT     hrPrev,
    ITcVnImage* ipSrcImage,
    ULONG       nXSrc,
    ULONG       nYSrc,
    ULONG       nWidth,
    ULONG       nHeight,
    ITcVnImage* ipDestImage,
    ULONG       nXDest,
    ULONG       nYDest
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
nXSrc	ULONG	Left boundary in ipSrcImage (inclusive 0-based index)
nYSrc	ULONG	Upper boundary in ipSrcImage (inclusive 0-based index)
nWidth	ULONG	ROI width
nHeight	ULONG	ROI height
ipDestImage	ITcVnImage* [▶ 1925]	Destination image (same type as ipSrcImage)
nXDest	ULONG	Left boundary in ipDestImage (inclusive 0-based index)
nYDest	ULONG	Upper boundary in ipDestImage (inclusive 0-based index)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.6 CopyIntoDisplayableImage

Copys an image into a displayable image. If you do not want to use ipSrcImage after this function call, you might want to use F_VN_TransformIntoDisplayableImage instead for better performance.

Syntax

Definition:

```
HRESULT CopyIntoDisplayableImage(
    HRESULT                 hrPrev,
    ITcVnImage*             ipSrcImage,
    ITcVnDisplayableImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnDisplayableImage* [▶ 1925]&	Returns the displayable image

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.7 CreateAssociatedImage

Creates a new image that shares its data with the source image. E.g. useful to work on different (disjoint) ROIs in parallel.

Syntax

Definition:

```
HRESULT CreateAssociatedImage(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Returns the created image (Non-zero interface pointers are reused.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.8 CreateEmptyImage

Creates an empty image without allocating any data buffer. The initial reference count is set to one if a new image interface is created and kept, otherwise.

Syntax

Definition:

```
HRESULT CreateEmptyImage(
    HRESULT      hrPrev,
    ITcVnImage*& ipImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]&	Returns the created image

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.9 CreateImage

Creates an image and allocate an appropriate data buffer. The initial reference count is set to one if a new image interface is created and kept, otherwise.

Syntax

Definition:

```
HRESULT CreateImage(
    HRESULT          hrPrev,
    ITcVnImage*&     ipImage,
    ULONG            nWidth,
    ULONG            nHeight,
    ETcVnElementType ePixelType,
    ULONG            nChannelNum
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]&	Returns the created image (Non-zero interface pointers are reused.)
nWidth	ULONG	Image width
nHeight	ULONG	Image height
ePixelType	ETcVnElementType [▶ 1738]	Pixel type
nChannelNum	ULONG	Number of channels (1 to 255)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.10 CreateImageAndSetPixels

Creates an image, allocates an appropriate data buffer and sets all pixels to the specified value. The initial reference count is set to one if a new image interface is created and kept, otherwise.

Syntax

Definition:

```
HRESULT CreateImageAndSetPixels(
    HRESULT            hrPrev,
    ITcVnImage*&       ipImage,
    ULONG              nWidth,
    ULONG              nHeight,
    ETcVnElementType   ePixelType,
    ULONG              nChannelNum,
    TcVnVector4_LREAL& aValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]&	Returns the created image (Non-zero interface pointers are reused.)
nWidth	ULONG	Image width
nHeight	ULONG	Image height
ePixelType	ETcVnElementType [▶ 1738]	Pixel type
nChannelNum	ULONG	Number of channels (1 to 4)
aValue	TcVnVector4_LREAL [▶ 1713]&	Pixel value

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.11 CreateImageFromArray

Creates an image and initialize it with the provided data. The initial reference count is set to one if a new image interface is created and kept, otherwise.

Syntax

Definition:

```
HRESULT CreateImageFromArray(
    HRESULT          hrPrev,
    PVOID            pData,
    ITcVnImage*&     ipImage,
    ULONG            nWidth,
    ULONG            nHeight,
    ETcVnElementType ePixelType,
    ULONG            nChannelNum
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
pData	PVOID	Pointer to the 1D data array. Make sure that the array contains at least nWidth * nHeight * nChannelNum elements and that the array element type matches ePixelType.
ipImage	ITcVnImage* [▶ 1925]&	Returns the created image (Non-zero interface pointers are reused.)
nWidth	ULONG	Image width
nHeight	ULONG	Image height
ePixelType	ETcVnElementType [▶ 1738]	Pixel type
nChannelNum	ULONG	Number of channels (1 to 255)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.12 ExportImage

Exports the image data into a given buffer (e.g. an array). F_VN_ExportImageSize should be called before to get the required buffer size.

Syntax

Definition:

```
HRESULT ExportImage(
    HRESULT     hrPrev,
    ITcVnImage* ipImage,
    PVOID       pBuffer,
    ULONGLONG   nBufferSize
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]	Image
pBuffer	PVOID	Pointer to a buffer to store the image data (Make sure to allocate enough memory! The required size in bytes can be determined using the function F_VN_ExportImageSize)
nBufferSize	ULONGLONG	Size of the buffer memory in bytes

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.13 ExportImageAsBmp

Exports the image as bitmap into a given buffer (e.g. an array). F_VN_ExportImageAsBmpSizeExp should be called before to get the required buffer size.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT ExportImageAsBmp(
    HRESULT     hrPrev,
    ITcVnImage* ipImage,
    PVOID       pBuffer,
    ULONGLONG   nBufferSize,
    ULONG       nWidth = 0,
    ULONG       nHeight = 0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]		Image
pBuffer	PVOID		Pointer to a buffer to store the bitmap image (Make sure to allocate enough memory! The required size in bytes can be determined using the function F_VN_ExportImageAsBmpSizeExp)
nBufferSize	ULONGLONG		Size of the buffer memory in bytes
nWidth	ULONG	0	Desired width (or 0 to keep the original width)
nHeight	ULONG	0	Desired height (or 0 to keep the original height)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.14 ExportImageAsBmpSize

Get the required buffer size (in bytes) to store the image as bitmap.

Syntax

Definition:

```
HRESULT ExportImageAsBmpSize(
    HRESULT     hrPrev,
    ITcVnImage* ipImage,
    ULONGLONG&  nBufferSize,
    ULONG       nWidth = 0,
    ULONG       nHeight = 0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]		Image
nBufferSize	ULONGLONG&		Returns the required buffer size in bytes
nWidth	ULONG	0	Desired width (or 0 to keep the original width)
nHeight	ULONG	0	Desired height (or 0 to keep the original height)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.15 ExportImageSize

Get the required buffer size (in bytes) to store the image data.

Syntax

Definition:

```
HRESULT ExportImageSize(
    HRESULT     hrPrev,
    ITcVnImage* ipImage,
    ULONGLONG&  nBufferSize
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]	Image
nBufferSize	ULONGLONG&	Returns the required buffer size in bytes

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.16 FuseImages

Fuse 2 images vertically (intended for line scan cameras).

Syntax

Definition:

```
HRESULT FuseImages(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage1,
    ITcVnImage*  ipSrcImage2,
    ITcVnImage*& ipDestImage,
    ULONG        nFirstLine,
    ULONG        nNumLines
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1925]	Source image 1
ipSrcImage2	ITcVnImage* [▶ 1925]	Source image 2
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image
nFirstLine	ULONG	Line index in ipSrcImage1, which is the first line in ipDestImage
nNumLines	ULONG	The number of lines that should be copied to ipDestImage, starting with nFirstLine in ipSrcImage1. Once the last line of ipSrcImage1 was copied, the remaining lines are copied from ipSrcImage2, starting at line index 0.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.17 FuseImagesArray

Fuse up to 10 images vertically (intended for line scan cameras).

Syntax

Definition:

```
HRESULT FuseImagesArray(
    HRESULT                 hrPrev,
    TcVnArray10_ITcVnImage& aSrcImageArray,
    ULONG                   nValidImages,
    ITcVnImage*&            ipDestImage,
    ULONG                   nFirstLine,
    ULONG                   nNumLines
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aSrcImageArray	TcVnArray10_ITcVnImage [▶ 1713]&	Ten-element source image array (not all elements need to be filled, nValidImages specifies the actual amount of images)
nValidImages	ULONG	Number of valid images in aSrcImageArray
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image
nFirstLine	ULONG	Line index in ipSrcImage1, which is the first line in ipDestImage
nNumLines	ULONG	The number of lines that should be copied to ipDestImage, starting with nFirstLine in aSrcImageArray[0]. Once the last line of aSrcImageArray[0] was copied, the remaining lines are copied from aSrcImageArray[1] (starting at line index 0) and so on.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.18 GetImageChannel

Return the specified source image channel as a single-channel image.

Syntax

Definition:

```
HRESULT GetImageChannel(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage,
    USHORT       nChannelIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate image will be created if required.)
nChannelIndex	USHORT	Index of the requested source image channel

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.19 GetImageHeight

Return the height of an image. (Alternatively use interface method .GetHeight.)

Syntax

Definition:

```
HRESULT GetImageHeight(
    HRESULT     hrPrev,
    ITcVnImage* ipImage,
    ULONG&      nHeight
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]	Image
nHeight	ULONG&	Return the height of ipImage

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.20 GetImageInfo

Gets a struct containing all common meta infos of the image. (Alternatively use interface method .GetImageInfo.)

Syntax

Definition:

```
HRESULT GetImageInfo(
    HRESULT        hrPrev,
    ITcVnImage*    ipImage,
    TcVnImageInfo& stImageInfo
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]	Image
stImageInfo	TcVnImageInfo [▶ 1768]&	Returns a struct describing the image

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.21 GetImageWidth

Return the width of an image. (Alternatively use interface method .GetWidth.)

Syntax

Definition:

```
HRESULT GetImageWidth(
    HRESULT     hrPrev,
    ITcVnImage* ipImage,
    ULONG&      nWidth
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]	Image
nWidth	ULONG&	Return the width of ipImage

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.22 GetPixel

Gets a specific pixel.

Syntax

Definition:

```
HRESULT GetPixel(
    HRESULT            hrPrev,
    ITcVnImage*        ipSrcImage,
    TcVnVector4_LREAL& aValue,
    ULONG              nX,
    ULONG              nY
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
aValue	TcVnVector4_LREAL [▶ 1713]&	Returns the pixel value (Unused channels are set to 0.)
nX	ULONG	x coordinate of the pixel
nY	ULONG	y coordinate of the pixel

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.23 GetPixelFormat

Gets the pixel format of an image. (Alternatively use interface method .GetPixelFormat.)

Syntax

Definition:

```
HRESULT GetPixelFormat(
    HRESULT          hrPrev,
    ITcVnImage*      ipImage,
    TcVnPixelFormat& stPixelFormat
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]	Image
stPixelFormat	TcVnPixelFormat [▶ 1780]&	Returns a struct describing the pixel format

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.24 GetRoi

Gets the coordinates of the region of interest (ROI) within the image.

Syntax

Definition:

```
HRESULT GetRoi(
    HRESULT              hrPrev,
    ITcVnImage*          ipSrcImage,
    TcVnRectangle_UDINT& stRoi
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
stRoi	TcVnRectangle_UDINT [▶ 1782]&	Returns the coordinates of the region of interest

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.25 MixImageChannels

Mix image channels by copying the specified channels of the source image into the specified channels of the destination image.

Syntax

Definition:

```
HRESULT MixImageChannels(
    HRESULT         hrPrev,
    ITcVnImage*     ipSrcImage,
    ITcVnImage*&    ipDestImage,
    ITcVnContainer* ipIndicesFromTo
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate image will be created if required.)
ipIndicesFromTo	ITcVnContainer* [▶ 1890]	Index pairs (ContainerType_Vector_TcVnVector2_DINT), specifying which source channel (TcVnVector2_DINT [0]) should be copied to which destination channel (TcVnVector2_DINT [1]).

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.26 ResetRoi

Reset the region of interest (ROI) of an image. (After this operation, the ROI is set to the entire image.)

Syntax

Definition:

```
HRESULT ResetRoi(
    HRESULT     hrPrev,
    ITcVnImage* ipImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]	Image

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.27 SetImageChannel

Sets the specified destination image channel to the values of the specified source image channel.

Syntax

Definition:

```
HRESULT SetImageChannel(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage,
    USHORT       nSrcChannelIndex,
    USHORT       nDestChannelIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate image will be created if required.)
nSrcChannelIndex	USHORT	Index of the source image channel
nDestChannelIndex	USHORT	Index of the destination image channel, which will be replaced by the specified source image channel

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.28 SetPixel

Sets a specific pixel.

Syntax

Definition:

```
HRESULT SetPixel(
    HRESULT             hrPrev,
    ITcVnImage*         ipSrcImage,
    TcVnVector4_LREAL&  aValue,
    ULONG               nX,
    ULONG               nY
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
aValue	TcVnVector4_LREAL [▶ 1713]&	The pixel value to set (additional channels are ignored.)
nX	ULONG	x coordinate of the pixel
nY	ULONG	y coordinate of the pixel

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.29 SetPixels

Sets all pixels of an image to a given value.

Syntax

Definition:

```
HRESULT SetPixels(
    HRESULT             hrPrev,
    ITcVnImage*         ipDestImage,
    TcVnVector4_LREAL&  aValue,
    ITcVnImage*         ipMask = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipDestImage	ITcVnImage* [▶ 1925]		Destination image
aValue	TcVnVector4_LREAL [▶ 1713]&		Pixel value
ipMask	ITcVnImage* [▶ 1925]	nullptr	Mask image

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.30 SetRoi

Sets a region of interest (ROI) within an image.

Syntax

Definition:

```
HRESULT SetRoi(
    HRESULT     hrPrev,
    ULONG       nX,
    ULONG       nY,
    ULONG       nWidth,
    ULONG       nHeight,
    ITcVnImage* ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nX	ULONG	Left boundary (inclusive 0-based index)
nY	ULONG	Upper boundary (inclusive 0-based index)
nWidth	ULONG	ROI width
nHeight	ULONG	ROI height
ipDestImage	ITcVnImage* [▶ 1925]	Destination image

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.31 SetRoi (TcVnRectangle_DINT)

Sets a region of interest (ROI) within an image.

Syntax

Definition:

```
HRESULT SetRoi(
    HRESULT             hrPrev,
    TcVnRectangle_DINT& stRoi,
    ITcVnImage*         ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stRoi	TcVnRectangle_DINT [▶ 1781]&	Region of interest
ipDestImage	ITcVnImage* [▶ 1925]	Destination image

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.32 SetRoi (TcVnRectangle_UDINT)

Sets a region of interest (ROI) within an image.

Syntax

Definition:

```
HRESULT SetRoi(
    HRESULT              hrPrev,
    TcVnRectangle_UDINT& stRoi,
    ITcVnImage*          ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stRoi	TcVnRectangle_UDINT [▶ 1782]&	Region of interest
ipDestImage	ITcVnImage* [▶ 1925]	Destination image

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.33 SplitImageChannels

Split a multi-channel image into multiple single channel images (1 for each source image channel).

Syntax

Definition:

```
HRESULT SplitImageChannels(
    HRESULT     hrPrev,
    ITcVnImage* ipSrcImage,
    PVOID       pDestImages,
    USHORT      nArraySize,
    USHORT&     nImageChannels
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image with multiple channels
pDestImages	PVOID	Pointer to an array of ITcVnImage (appropriate destination images will be created if required)
nArraySize	USHORT	Number of pDestImages array elements (array must be >= ipSourceImage channels)
nImageChannels	USHORT&	Actual source image channels

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.34 TransformIntoDisplayableImage

Transform an image into a displayable image. The source image will be released and zeroed while existing destination images will be released and overwritten with the function result. If bAllowDeepCopy equals false, the source image must not be used anywhere else and the transformation is always very fast. Otherwise, if bAllowDeepCopy equals true, a deep copy of the source image might be created if required, which will result in a longer execution time. If you want to use ipSrcImage after this function call, use F_VN_CopyIntoDisplayableImage instead.

Syntax

Definition:

```
HRESULT TransformIntoDisplayableImage(
    HRESULT                 hrPrev,
    ITcVnImage*&            ipSrcImage,
    ITcVnDisplayableImage*& ipDestImage,
    bool                    bAllowDeepCopy = false
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]&		Source image
ipDestImage	ITcVnDisplayableImage* [▶ 1925]&		Returns the displayable image
bAllowDeepCopy	bool	false	Allow deep image copies, if required

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.35 CreateNDimensionalImage

Creates an N-dimensional image and allocates an appropriate data buffer. The initial reference count is set to one if a new image interface is created and kept, otherwise.

Syntax

Definition:

```
HRESULT CreateNDimensionalImage(
    HRESULT            hrPrev,
    ITcVnImage*&       ipImage,
    TcVnArray33_UDINT& aShape,
    ETcVnElementType   ePixelType
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]&	Returns the created image (Non-zero interface pointers are reused.)
aShape	TcVnArray33_UDINT [▶ 1713]&	Specifies the shape of ipDestImage. First entry is the number of dimensions of ipImage (max. 32), followed by the sizes of each dimension.
ePixelType	ETcVnElementType [▶ 1738]	Pixel type

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.36 CreateNDimensionalImageAndSetPixels

Creates an N-dimensional image, allocates an appropriate data buffer and sets all pixels to the specified value. The initial reference count is set to one if a new image interface is created and kept, otherwise.

Syntax

Definition:

```
HRESULT CreateNDimensionalImageAndSetPixels(
    HRESULT            hrPrev,
    ITcVnImage*&       ipImage,
    TcVnArray33_UDINT& aShape,
    ETcVnElementType   ePixelType,
    double             fValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]&	Returns the created image (Non-zero interface pointers are reused.)
aShape	TcVnArray33_UDINT [▶ 1713]&	Specifies the shape of ipDestImage. First entry is the number of dimensions of ipImage (max. 32), followed by the sizes of each dimension.
ePixelType	ETcVnElementType [▶ 1738]	Pixel type
fValue	double	Pixel value

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.37 CreateNDimensionalImageFromArray

Creates an N-dimensional image and initialize it with the provided data. The initial reference count is set to one if a new image interface is created and kept, otherwise.

Syntax

Definition:

```
HRESULT CreateNDimensionalImageFromArray(
    HRESULT            hrPrev,
    PVOID              pData,
    ITcVnImage*&       ipImage,
    TcVnArray33_UDINT& aShape,
    ETcVnElementType   ePixelType
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
pData	PVOID	Pointer to the 1D data array. Ensure that the array contains at least the number of elements equal to the product of all dimensions in aShape and that the array element type matches ePixelType.
ipImage	ITcVnImage* [▶ 1925]&	Returns the created image (Non-zero interface pointers are reused.)
aShape	TcVnArray33_UDINT [▶ 1713]&	Specifies the shape of ipDestImage. First entry is the number of dimensions of ipImage (max. 32), followed by the sizes of each dimension.
ePixelType	ETcVnElementType [▶ 1738]	Pixel type

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.38 GetLayoutInfo

Return the layout information of an image

Syntax

Definition:

```
HRESULT GetLayoutInfo(
    HRESULT            hrPrev,
    ITcVnImage*        ipImage,
    TcVnArray33_UDINT& aLayoutInfo
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]	Source image
aLayoutInfo	TcVnArray33_UDINT [▶ 1713]&	Contains layout information of ipImage. First entry of the aLayoutInfo is the number of dimensions of ipImage (max. 32), followed by the sizes of each dimension.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.39 PadImageBorder

Add padding around the original image borders.

Syntax

Definition:

```
HRESULT PadImageBorder(
    HRESULT                          hrPrev,
    ITcVnImage*                      ipSrcImage,
    ITcVnImage*&                     ipPaddedImage,
    ULONG                            nTopBorder,
    ULONG                            nBottomBorder,
    ULONG                            nLeftBorder,
    ULONG                            nRightBorder,
    ETcVnBorderInterpolationMethod ePaddingType = BIM_CONSTANT,
    TcVnVector4_LREAL&               aPaddingValue = {0, 0, 0, 0}
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipPaddedImage	ITcVnImage* [▶ 1925]&		Padded destination image (Same type as ipSrcImage, an appropriate destination image will be created if required.)
nTopBorder	ULONG		Padding height in pixels above top border
nBottomBorder	ULONG		Padding height in pixels below bottom border
nLeftBorder	ULONG		Padding width in pixels before left border
nRightBorder	ULONG		Padding width in pixels after right border
ePaddingType	ETcVnBorderInterpolationMethod [▶ 1716]	BIM_CONSTANT	Specifies how the pixel values of the padding area are determined
aPaddingValue	TcVnVector4_LREAL [▶ 1713]&	{0, 0, 0, 0}	Specifies the padding value if CONSTANT is used

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.5.40 ReinterpretUnsupportedImage

Reinterpret an unsupported image using additional information. If successful, the image will be of a supported type.

Syntax

Definition:

```
HRESULT ReinterpretUnsupportedImage(
    HRESULT          hrPrev,
    ITcVnImage*      ipImage,
    ETcVnElementType ePixelType
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]	Image to be reinterpreted
ePixelType	ETcVnElementType [▶ 1738]	Pixel element type to be used for reinterpreting the image data

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.6 Code Quality

Diese Gruppe enthält Funktionen zur Qualitätsbewertung von verschiedenen 1D- und 2D-Codes. Die Bewertung erfolgt nach internationalen Standards. Die Prüfung der Codequalität soll sicherstellen, dass der entsprechende Code zum Zeitpunkt der Verwendung zuverlässig dekodiert werden kann.

Um vergleichbare Ergebnisse zu erhalten, definieren die Standards Methoden zur Messung und Bewertung von spezifischen Attributen sowie zur Ableitung einer Gesamtbewertung der Codequalität. Das Ziel ist, die zu erwartende Lesbarkeit der Codes, unter Verwendung von verschiedenen Arten von Lesegeräten, in verschiedenen Anwendungen, Umgebungen und unter unterschiedlichen Bedingungen weitestgehend vorherzusagen. Üblicherweise wird daher für jede Anwendung einzeln eine minimale Qualitätsstufe festgelegt, die akzeptabel ist.

Zum einen beschreiben die Standards Anforderungen, die von den verwendeten Systemen eingehalten werden müssen, um überhaupt verwertbare Ergebnisse zu erhalten. Zum anderen geben die Standards Empfehlungen wie z.B. zur Beleuchtung, zu Objektiveinstellungen und zu dem geometrischen Aufbau, um die Bewertungen mit anderen Systemen vergleichen zu können. Wenn man das Ziel verfolgt, ein System aufzubauen, das dem Standard entspricht, empfiehlt es sich, die entsprechenden Spezifikationen zur Qualitätsbewertung und den zu prüfenden Codes zu beschaffen.

Die Funktionen können auch mit Systemen verwendet werden, die nicht dem Standard entsprechen. Dies hat nur zur Folge, dass die ermittelte Qualität nicht mit anderen Systemen verglichen werden kann. Generell sind kleinere Unterschiede, in den einzelnen Bewertungskategorien, bei unterschiedlichen Systemen zu erwarten.

Nach dem Standard ISO/IEC 15416:2016 kann die Druckqualität von Strichcodes für die Codetypen Code 39, Code 128, EAN 8, EAN 13, ITF, UPC-A, UPC-E mit den folgenden Funktionen ermittelt werden:

- GradeBarcode [▶ 2259]

Nach dem Standard ISO/IEC 15415:2011 kann die Druckqualität von zweidimensionalen Matrixsymbolen der Typen ECC200 Data Matrix-Code und QR-Code mit den folgenden Funktionen ermittelt werden:

- GradeDataMatrixCode [▶ 2261]
- GradeQRCode [▶ 2263]

Interpretation des HRESULT

Für die Code Grading Funktionen werden die HRESULT [▶ 150]–Rückgabewerte wie folgt benutzt:

Code	Name	Beschreibung
0x000	S_OK	Funktion wurde erfolgreich ausgeführt und alle Bewertungen konnten entsprechend dem Standard durchgeführt werden.
0x001	S_FALSE	Funktion wurde erfolgreich ausgeführt, bei 1D Codes konnten nicht alle Scanlinien dekodiert werden und bei 2D-Codes konnten nicht alle Prüfungen dem Standard entsprechend durchgeführt werden. Die Gesamtnote wurde daher auf 0 gesetzt, die einzelnen Ergebnisse können aber ausgewertet werden.
0x256	S_WATCHDOGTIMEOUT	Funktion wurde vom Watchdog abgebrochen. Je nach Zeitpunkt des Abbruchs können einzelne Ergebnisse ausgewertet werden
0x70C	NOTFOUND	Funktion wurde nicht erfolgreich ausgeführt, bei 1D Codes konnte keine Scanlinie dekodiert werden und bei 2D Codes wurde der Code nicht gefunden bzw. konnte nicht dekodiert werden.
0x7xx	Sämtliche Error-Codes	Funktion wurde nicht erfolgreich ausgeführt. Siehe dazu: ADS Return Codes [▶ 2966]

6.2.3.6.1 GradeBarcode

Grades a 1D barcode according to ISO / IEC 15416:2016. Requires uniform illumination, good focus and the image sensor plane parallel to the code plane to achieve meaningful results.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT GradeBarcode(
    HRESULT                     hrPrev,
    ITcVnImage*                 ipSrcImage,
    ITcVnContainer*&            ipDecodedData,
    TcVnCodeGrades1D&           stCodeGrades,
    ETcVnBarcodeType            eBarcodeType,
    float&                      fAngleDeg,
    ETcVnBarcodeSearchDirection eSearchDirection = BSD_ANY,
    ITcVnImage**                pipResultVisualization = nullptr,
    ITcVnContainer**            pipScanLineInfo = nullptr,
    ITcVnContainer**            pipContour = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image, only containing the horizontally aligned barcode region (USINT, 1 channel, including quiet zones on left and right, excluding Text or empty space on more than 10% total height on top or bottom)
ipDecodedData	ITcVnContainer* [▶ 1890]&		Returns the decoded data (ContainerType_String_SINT)
stCodeGrades	TcVnCodeGrades1D [▶ 1766]&		Returns the code grades in the range from 0 (very bad) to 4 (very good)
eBarcodeType	ETcVnBarcodeType [▶ 1715]		Type of the barcode (supported: CODE39, CODE128, EAN8, EAN13, ITF, UPCA, UPCE)
fAngleDeg	float&		Returns the approximate clockwise rotation angle in degree
eSearchDirection	ETcVnBarcodeSearchDirection [▶ 1714]	BSD_ANY	Barcode search direction (BSD_ANY first tries horizontal, then vertical)
pipResultVisualization	ITcVnImage* [▶ 1925]*	nullptr	Returns a visualization of the scan lines (optional, set to 0 if not required). Grades are color coded: >= 3 as green, >= 2 as light blue, >= 1 as orange, < 1 as red
pipScanLineInfo	ITcVnContainer* [▶ 1890]*	nullptr	Returns additional scan line info for further evaluation, e.g. to use PlotIntensityProfile for visualization (optional, set to 0 if not required; ContainerType_Vector_TcVnVector3_REAL, containing [row index, overall grade, threshold])
pipContour	ITcVnContainer* [▶ 1890]*	nullptr	Returns the 4 approximate code corner points, including the quiet zone (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_DINT)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Code Quality

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.6.2 GradeDataMatrixCode

Grades a Data Matrix code (ECC200) according to ISO / IEC 15415:2011. Requires uniform illumination, good focus and the image sensor plane parallel to the code plane to achieve meaningful results.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT GradeDataMatrixCode(
    HRESULT           hrPrev,
    ITcVnImage*       ipSrcImage,
    ITcVnContainer*&  ipDecodedData,
    TcVnCodeGradesDM& stCodeGrades,
    float             fModuleWidth,
    float&            fAngleDeg,
    ITcVnImage**      pipResultVisualization = nullptr,
    ITcVnContainer**  pipMarginGrades = nullptr,
    ITcVnContainer**  pipContour = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image, only containing the Data Matrix code region (USINT, 1 channel, including quiet zone)
ipDecodedData	ITcVnContainer* [▶ 1890]&		Returns the decoded data (ContainerType_String_SINT)
stCodeGrades	TcVnCodeGradesDM [▶ 1766]&		Returns the code grades in the range from 0 (very bad) to 4 (very good)
fModuleWidth	float		Minimum module width of the code in the image (in pixels). Must be at least 3, preferably 5 - 8.
fAngleDeg	float&		Returns the clockwise rotation angle in degree
pipResultVisualization	ITcVnImage* [▶ 1925]*	nullptr	Returns a visualization of the scan grid (optional, set to 0 if not required). Module margin grades are color coded: >= 3 as green, >= 2 as light blue, >= 1 as orange, < 1 as red
pipMarginGrades	ITcVnContainer* [▶ 1890]*	nullptr	Returns the individual margin grade for each module, e.g. to create a custom visualization (optional, set to 0 if not required; ContainerType_Vector_TcVnVector3_REAL, containing [x-position, y-position, grade])
pipContour	ITcVnContainer* [▶ 1890]*	nullptr	Returns the 4 code corner points, excluding the quiet zone (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_DINT)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Code Quality

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.6.3 GradeQRCode

Grades a QR code according to ISO / IEC 15415:2011. Requires uniform illumination, good focus and the image sensor plane parallel to the code plane to achieve meaningful results.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT GradeQRCode(
    HRESULT           hrPrev,
    ITcVnImage*       ipSrcImage,
    ITcVnContainer*&  ipDecodedData,
    TcVnCodeGradesQR& stCodeGrades,
    float             fModuleWidth,
    float&            fAngleDeg,
    ITcVnImage**      pipResultVisualization = nullptr,
    ITcVnContainer**  pipMarginGrades = nullptr,
    ITcVnContainer**  pipContour = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image, only containing the QR code region (USINT, 1 channel, including quiet zone)
ipDecodedData	ITcVnContainer* [▶ 1890]&		Returns the decoded data (ContainerType_String_SINT)
stCodeGrades	TcVnCodeGradesQR [▶ 1767]&		Returns the code grades in the range from 0 (very bad) to 4 (very good)
fModuleWidth	float		Minimum module width of the code in the image (in pixels). Must be at least 3, preferably 5 - 8.
fAngleDeg	float&		Returns the clockwise rotation angle in degree
pipResultVisualization	ITcVnImage* [▶ 1925]*	nullptr	Returns a visualization of the scan grid (optional, set to 0 if not required). Module margin grades are color coded: >= 3 as green, >= 2 as light blue, >= 1 as orange, < 1 as red
pipMarginGrades	ITcVnContainer* [▶ 1890]*	nullptr	Returns the individual margin grade for each module, e.g. to create a custom visualization (optional, set to 0 if not required; ContainerType_Vector_TcVnVector3_REAL, containing [x-position, y-position, grade])
pipContour	ITcVnContainer* [▶ 1890]*	nullptr	Returns the 4 code corner points, excluding the quiet zone (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_DINT)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Code Quality

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.7 Code Reading

Diese Gruppe enthält Funktionen zur Detektion und zum Lesen von 1D- und 2D-Codes.

Funktionen

1D Codes

- ReadBarcode(Exp) [▶ 2268]
  - CodaBar
  - Code39
  - Code93
  - Code128
  - EAN8
  - EAN13
  - ITF
  - UPCA
  - UPCE
  - Code39Extended
- ReadPharmaCode [▶ 2271]

2D Codes

- ReadDataMatrixCode [▶ 2270]
- ReadQRCode [▶ 2272]
- ReadDataMatrixCodeRoi [▶ 2274]

Code-Anforderungen

Grundsätzlich wird eine Modulgröße von mindestens drei Pixeln vorausgesetzt; für stabilere Ergebnisse werden mehr als drei Pixel empfohlen. Das Bild sollte scharf sein und einen hohen Kontrast haben. Zudem wird eine Codeausrichtung von ca. 0° oder 90° empfohlen, sodass der Code parallel zu einer Bildachse liegt.

Interpretation des HRESULT

Für die Code Reading Funktionen werden die HRESULT [▶ 150]–Rückgabewerte wie folgt benutzt:

Code	Name	Beschreibung
0x000	S_OK	Funktion wurde erfolgreich ausgeführt und alle erwarteten Codes wurden gefunden (mindestens ein Code bei den Standard-Funktionen und nCodeNumber bei den Experten-Funktionen).
0x001	S_FALSE	Funktion wurde erfolgreich ausgeführt, aber nicht alle erwarteten Codes wurden gefunden (mindestens ein Code bei den Standard-Funktionen und nCodeNumber bei den Experten-Funktionen).
0x256	S_WATCHDOGTIMEOUT	Funktion wurde vom Watchdog abgebrochen. Einige Code Reading Funktionen können Teilergebnisse zurückgeben (siehe dazu einzelne Funktionsbeschreibungen).
0x7xx	Sämtliche Error-Codes	Funktion wurde nicht erfolgreich ausgeführt. Siehe dazu: ADS Return Codes [▶ 2966]

Daher genügt die übliche Überprüfung mit `SUCCEEDED()` nicht, um auf einen gefundenen Code und damit auf vorhandene Daten in `ipDecodedData` zu schließen. Dazu kann folgende Abfrage benutzt werden:

```
if (hr == S_OK)
{
    // Export Code into String
    hr = ExportSubContainer_String(hr, ipDecodedCode, 0, sCodeAsString, 255);
    // Use sCodeAsString
}
else if (SUCCEEDED(hr))
{
    // Process partial results
}
else
{
    // Error handling
}
```

Suchstrategien bei 2D-Codes

Für die 2D-Codes werden die Suchalgorithmen auf verschiedene Varianten des Eingangsbildes angewandt, um möglichst viele Codes zu erkennen. Zum Generieren verschiedener Varianten wird Invertieren sowie Spiegeln des Bildes genutzt. Welche Varianten ausprobiert werden, wird als Suchstrategie bezeichnet. Damit Sie mehr Kontrolle über die Laufzeit der Funktionen haben, kann diese Suchstrategie konfiguriert werden. Dafür steht das Enum ETcVn2dCodeSearchStrategy [▶ 1713] zur Verfügung. Damit kann für jede mögliche Bildtransformation (Invertierung und Spiegelung) jeweils festgelegt werden, ob sie angewandt werden soll und ob das originale Bild oder das transformierte Bild als erstes auf Codes überprüft werden soll.

Der Funktionsparameter `eSearchStrategy` definiert für entsprechende Code Reading Expertenfunktionen genau diese Suchstrategie. Die folgenden Tabellen zeigen die verschiedenen Kombinationen:

Tab. 3: Invertierung

eSearchStrategy	Originalbild	Invertiertes Bild
CSS_ONLY_NOT_INVERTED	1.	-
CSS_FIRST_NOT_INVERTED	1.	2.
CSS_ONLY_INVERTED	-	1.
CSS_FIRST_INVERTED	2.	1.

Tab. 4: Spiegelung

eSearchStrategy	Originalbild	Gespiegeltes Bild
CSS_ONLY_NOT_FLIPPED	1.	-
CSS_FIRST_NOT_FLIPPED	1.	2.
CSS_ONLY_FLIPPED	-	1.
CSS_FIRST_FLIPPED	2.	1.

Für jede Transformationsart kann eine Suchstrategie festgelegt werden. Die Suchstrategien der verschiedenen Suchstrategien können wie folgt verknüpft werden:

```
eSearchStrategy = CSS_ONLY_FLIPPED | CSS_FIRST_INVERTED;
```

Zudem kann mit der Option `CSS_DEFAULT` eine Standard-Einstellung ausgewählt werden, die je nach Code-Art variiert.

Verknüpfung von Suchstrategien nicht möglich

Eine Verknüpfung mehrerer Suchstrategien derselben Transformationsart ist nicht zulässig. Es kann jeweils nur eine Strategie für Invertierung und Spiegelung miteinander verknüpft werden.

6.2.3.7.1 ReadBarcode

Detect and interpret a 1d barcode within the provided image. Can be canceled by Watchdog.

Syntax

Definition:

```
HRESULT ReadBarcode(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage,
    ITcVnContainer*& ipDecodedData,
    ULONG            eBarcodeType
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (USINT elements, 1 channel or 3 channel (3 channel input is expected to be RGB and internally converted to Gray))
ipDecodedData	ITcVnContainer* [▶ 1890]&	Returns the decoded code (ContainerType_Vector_String_SINT)
eBarcodeType	ULONG	Types of barcode to search for (ETcVnBarcodeType)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Code Reading

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.7.2 ReadBarcode (Exp)

Detect and interpret a 1d barcode within the provided image. Can be canceled by Watchdog.

Syntax

Definition:

```
HRESULT ReadBarcode(
    HRESULT                     hrPrev,
    ITcVnImage*                 ipSrcImage,
    ITcVnContainer*&            ipDecodedData,
    ITcVnContainer*&            ipContours,
    ULONG                       eBarcodeType,
    LONG                        nCodeNumber,
    ETcVnBarcodeSearchDirection eSearchDirection,
    ITcVnContainer**            pipAngles = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (USINT elements, 1 channel or 3 channel (3 channel input is expected to be RGB and internally converted to Gray))
ipDecodedData	ITcVnContainer* [▶ 1890]&		Returns the decoded code (ContainerType_Vector_String_SINT)
ipContours	ITcVnContainer* [▶ 1890]&		Returns the code positions as contours (optional, set to 0 if not required; ContainerType_Vector_Vector_TcVnPoint2_DINT)
eBarcodeType	ULONG		Types of barcode to search for (ETcVnBarcodeType)
nCodeNumber	LONG		Number of 1d barcode that should be detected within the provided image (currently only 1 supported).
eSearchDirection	ETcVnBarcodeSearchDirection [▶ 1714]		Barcode search direction.
pipAngles	ITcVnContainer* [▶ 1890]*	nullptr	Returns the clockwise code rotation angles (optional, set to 0 if not required; ContainerType_Vector_REAL)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Code Reading

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.7.3 ReadDataMatrixCode

Detect and interpret a data matrix code (ECC200) within the provided image.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT ReadDataMatrixCode(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage,
    ITcVnContainer*& ipDecodedData,
    ITcVnContainer** pipContours = nullptr,
    LONG             nCodeNumber = 1,
    ULONG            eSearchStrategy = CSS_DEFAULT,
    ITcVnContainer** pipAngles = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (USINT elements, 1 or 3 channels)
ipDecodedData	ITcVnContainer* [▶ 1890]&		Returns the decoded code (ContainerType_Vector_String_SINT)
pipContours	ITcVnContainer* [▶ 1890]*	nullptr	Returns the code positions as contours (optional, set to 0 if not required; ContainerType_Vector_Vector_TcVnPoint2_DINT)
nCodeNumber	LONG	1	Number of data matrix codes that should be detected within the provided image. (If set to -1, it tries to detect all data matrix codes.)
eSearchStrategy	ULONG	CSS_DEFAULT	Used search strategy (ETcVn2dCodeSearchStrategy)
pipAngles	ITcVnContainer* [▶ 1890]*	nullptr	Returns the clockwise code rotation angles (optional, set to 0 if not required; ContainerType_Vector_REAL)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Code Reading

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.7.4 ReadPharmaCode

Detect and interpret a pharma code within the provided image. Can be canceled by Watchdog.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT ReadPharmaCode(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage,
    ITcVnContainer*& ipDecodedData,
    ITcVnContainer** pipContours = nullptr,
    LONG             nCodeNumber = 1,
    SHORT            nMinBarNumber = 4,
    ITcVnContainer** pipAngles = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (1 channel (all element types) or 3 channel with elements of type TCVN_ET_USINT, TCVN_ET_UINT or TCVN_ET_REAL (3 channel input is expected to be RGB and internally converted to Gray))
ipDecodedData	ITcVnContainer* [▶ 1890]&		Returns the decoded code (ContainerType_Vector_String_SINT)
pipContours	ITcVnContainer* [▶ 1890]*	nullptr	Returns the code positions as contours (optional, set to 0 if not required; ContainerType_Vector_Vector_TcVnPoint2_DINT)
nCodeNumber	LONG	1	Number of pharma codes that should be detected within the provided image (currently only 1 supported).
nMinBarNumber	SHORT	4	Minimal number of (dark) bars that codes must have.
pipAngles	ITcVnContainer* [▶ 1890]*	nullptr	Returns the clockwise code rotation angles (optional, set to 0 if not required; ContainerType_Vector_REAL)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Code Reading

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.7.5 ReadQRCode

Detect and interpret a QR code within the provided image. Can be canceled by Watchdog.

Syntax

Definition:

```
HRESULT ReadQRCode(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage,
    ITcVnContainer*& ipDecodedData,
    ITcVnContainer** pipContours = nullptr,
    LONG             nCodeNumber = 1,
    ULONG            eSearchStrategy = CSS_DEFAULT,
    ITcVnContainer** pipAngles = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (USINT elements, 1 channel or 3 channel (3 channel input is expected to be RGB and internally converted to Gray))
ipDecodedData	ITcVnContainer* [▶ 1890]&		Returns the decoded code (ContainerType_Vector_String_SINT)
pipContours	ITcVnContainer* [▶ 1890]*	nullptr	Returns the code positions as contours (optional, set to 0 if not required; ContainerType_Vector_Vector_TcVnPoint2_DINT)
nCodeNumber	LONG	1	Number of QR codes that should be detected within the provided image (currently only 1 supported).
eSearchStrategy	ULONG	CSS_DEFAULT	Used search strategy (ETcVn2dCodeSearchStrategy)
pipAngles	ITcVnContainer* [▶ 1890]*	nullptr	Returns the clockwise code rotation angles (optional, set to 0 if not required; ContainerType_Vector_REAL)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Code Reading

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.7.6 ReadDataMatrixCodeRoi

Detect and interpret a data matrix code (ECC200) within the provided image. The code should be in the image center.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT ReadDataMatrixCodeRoi(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage,
    ITcVnContainer*& ipDecodedData,
    float            fModuleWidth,
    float&           fAngleDeg,
    ITcVnContainer** pipContour = nullptr,
    ULONGLONG        eOptions = DMO_DEFAULT,
    unsigned char    nRows = 0,
    unsigned char    nCols = 0,
    bool             bExtensiveSearch = true
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (USINT elements, 1 channel)
ipDecodedData	ITcVnContainer* [▶ 1890]&		Returns the decoded code (ContainerType_String_SINT)
fModuleWidth	float		Minimum module width of the code in the image (in pixels). Must be at least 3, preferably 5 - 8.
fAngleDeg	float&		Returns the clockwise rotation angle in degree
pipContour	ITcVnContainer* [▶ 1890]*	nullptr	Returns the contour of the detected code (ContainerType_Vector_TcVnPoint2_DINT; optional, set to 0 if not required)
eOptions	ULONGLONG	DMO_DEFAULT	Specifies a combination of one or more ETcVnDataMatrixOptions
nRows	unsigned char	0	Specifies the number of code module rows, including the finder and clock pattern, excluding the quiet zone (8 - 144; set to 0 to search for any size)
nCols	unsigned char	0	Specifies the number of code module columns, including the finder and clock pattern, excluding the quiet zone (10 - 144; set to 0 to search for any size)
bExtensiveSearch	bool	true	If true, tries different approaches if the specified options fail

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Code Reading

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8 Container Statistics

Diese Gruppe enthält Funktionen zur statistischen Analyse von Containern [▶ 163].

Funktionen

Mit euklidischer Norm

- ContainerAverage [▶ 2340]
- ContainerAverageVariance [▶ 2343]
- MaxElement [▶ 2346]
- MedianElement [▶ 2347]
- MinElement [▶ 2348]

Elementweise

- ContainerAverageElementwise [▶ 2341]
- ContainerAverageVarianceElementwise [▶ 2344]
- MaxElementElementwise [▶ 2276]
- MedianElementElementwise [▶ 2298]
- MinElementElementwise [▶ 2319]

6.2.3.8.1 MaxElementElementwise

Dieses Kapitel enthält Funktionen, mit denen der Max-Wert entsprechend dem Datentyp Elementweise ermittelt wird. Diese Funktion ist für Container mit Grundelementen verfügbar.

Z. B. bei einem TcVnPoint2_DINT werden der maximale X-Wert und der maximale Y-Wert unabhängig voneinander ermittelt, daher müssen die Werte nicht von dem gleichen Punkt stammen.

6.2.3.8.1.1 MaxElementElementwise (LONG)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    LONG&           nMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
nMax	LONG&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.2 MaxElementElementwise (SHORT)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    SHORT&          nMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
nMax	SHORT&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.3 MaxElementElementwise (double)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    double&         fMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
fMax	double&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.4 MaxElementElementwise (float)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    float&          fMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
fMax	float&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.5 MaxElementElementwise (char)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    char&           nMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
nMax	char&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.6 MaxElementElementwise (TcVnPoint2_DINT)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnPoint2_DINT& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnPoint2_DINT [▶ 1711]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.7 MaxElementElementwise (TcVnPoint2_LREAL)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnPoint2_LREAL& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnPoint2_LREAL [▶ 1711]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.8 MaxElementElementwise (TcVnPoint2_REAL)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnPoint2_REAL& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnPoint2_REAL [▶ 1711]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.9 MaxElementElementwise (TcVnPoint3_LREAL)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnPoint3_LREAL& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnPoint3_LREAL [▶ 1711]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.10 MaxElementElementwise (TcVnPoint3_REAL)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnPoint3_REAL& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnPoint3_REAL [▶ 1711]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.11 MaxElementElementwise (TcVnVector2_DINT)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_DINT& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnVector2_DINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.12 MaxElementElementwise (TcVnVector2_INT)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnVector2_INT& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnVector2_INT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.13 MaxElementElementwise (TcVnVector2_REAL)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_REAL& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnVector2_REAL [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.14 MaxElementElementwise (TcVnVector2_SINT)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_SINT& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnVector2_SINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.15 MaxElementElementwise (TcVnVector2_UINT)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_UINT& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnVector2_UINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.16 MaxElementElementwise (TcVnVector2_USINT)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContainer,
    TcVnVector2_USINT& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnVector2_USINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.17 MaxElementElementwise (TcVnVector3_INT)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnVector3_INT& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnVector3_INT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.18 MaxElementElementwise (TcVnVector3_REAL)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector3_REAL& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnVector3_REAL [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.19 MaxElementElementwise (TcVnVector3_SINT)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector3_SINT& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnVector3_SINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.20 MaxElementElementwise (TcVnVector3_UINT)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector3_UINT& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnVector3_UINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.21 MaxElementElementwise (TcVnVector3_USINT)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContainer,
    TcVnVector3_USINT& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnVector3_USINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.22 MaxElementElementwise (TcVnVector4_DINT)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector4_DINT& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnVector4_DINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.23 MaxElementElementwise (TcVnVector4_INT)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnVector4_INT& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnVector4_INT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.24 MaxElementElementwise (TcVnVector4_LREAL)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContainer,
    TcVnVector4_LREAL& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnVector4_LREAL [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.25 MaxElementElementwise (TcVnVector4_SINT)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector4_SINT& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnVector4_SINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.26 MaxElementElementwise (TcVnVector4_UINT)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector4_UINT& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnVector4_UINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.27 MaxElementElementwise (TcVnVector4_USINT)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContainer,
    TcVnVector4_USINT& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnVector4_USINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.28 MaxElementElementwise (ULONG)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    ULONG&          nMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
nMax	ULONG&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.29 MaxElementElementwise (USHORT)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    USHORT&         nMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
nMax	USHORT&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.30 MaxElementElementwise (ULONGLONG)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    ULONGLONG&      nMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
nMax	ULONGLONG&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.31 MaxElementElementwise (unsigned char)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    unsigned char&  nMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
nMax	unsigned char&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.1.32 MaxElementElementwise (TcVnVector4_REAL)

Gets the element wise maximum container element.

Syntax

Definition:

```
HRESULT MaxElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector4_REAL& aMax
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMax	TcVnVector4_REAL [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2 MedianElementElementwise

Dieses Kapitel enthält Funktionen, mit denen der Median-Wert entsprechend dem Datentyp Elementweise ermittelt wird. Diese Funktion ist für Container mit Grundelementen verfügbar.

Z. B. bei einem TcVnPoint2_DINT werden der median-X-Wert und der median-Y-Wert unabhängig voneinander ermittelt, daher müssen die Werte nicht von dem gleichen Punkt stammen.

6.2.3.8.2.1 MedianElementElementwise (LONG)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    LONG&           nMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
nMedian	LONG&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.2 MedianElementElementwise (SHORT)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    SHORT&          nMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
nMedian	SHORT&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.3 MedianElementElementwise (double)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    double&         fMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
fMedian	double&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.4 MedianElementElementwise (float)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    float&          fMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
fMedian	float&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.5 MedianElementElementwise (char)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    char&           nMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
nMedian	char&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.6 MedianElementElementwise (TcVnPoint2_DINT)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnPoint2_DINT& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnPoint2_DINT [▶ 1711]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.7 MedianElementElementwise (TcVnPoint2_LREAL)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnPoint2_LREAL& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnPoint2_LREAL [▶ 1711]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.8 MedianElementElementwise (TcVnPoint2_REAL)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnPoint2_REAL& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnPoint2_REAL [▶ 1711]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.9 MedianElementElementwise (TcVnPoint3_LREAL)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnPoint3_LREAL& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnPoint3_LREAL [▶ 1711]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.10 MedianElementElementwise (TcVnPoint3_REAL)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnPoint3_REAL& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnPoint3_REAL [▶ 1711]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.11 MedianElementElementwise (TcVnVector2_DINT)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_DINT& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnVector2_DINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.12 MedianElementElementwise (TcVnVector2_INT)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnVector2_INT& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnVector2_INT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.13 MedianElementElementwise (TcVnVector2_REAL)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_REAL& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnVector2_REAL [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.14 MedianElementElementwise (TcVnVector2_SINT)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_SINT& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnVector2_SINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.15 MedianElementElementwise (TcVnVector2_UINT)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_UINT& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnVector2_UINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.16 MedianElementElementwise (TcVnVector2_USINT)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContainer,
    TcVnVector2_USINT& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnVector2_USINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.17 MedianElementElementwise (TcVnVector3_INT)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnVector3_INT& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnVector3_INT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.18 MedianElementElementwise (TcVnVector3_REAL)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector3_REAL& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnVector3_REAL [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.19 MedianElementElementwise (TcVnVector3_SINT)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector3_SINT& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnVector3_SINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.20 MedianElementElementwise (TcVnVector3_UINT)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector3_UINT& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnVector3_UINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.21 MedianElementElementwise (TcVnVector3_USINT)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContainer,
    TcVnVector3_USINT& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnVector3_USINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.22 MedianElementElementwise (TcVnVector4_DINT)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector4_DINT& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnVector4_DINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.23 MedianElementElementwise (TcVnVector4_INT)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnVector4_INT& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnVector4_INT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.24 MedianElementElementwise (TcVnVector4_LREAL)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContainer,
    TcVnVector4_LREAL& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnVector4_LREAL [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.25 MedianElementElementwise (TcVnVector4_SINT)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector4_SINT& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnVector4_SINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.26 MedianElementElementwise (TcVnVector4_UINT)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector4_UINT& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnVector4_UINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.27 MedianElementElementwise (TcVnVector4_USINT)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContainer,
    TcVnVector4_USINT& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnVector4_USINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.28 MedianElementElementwise (ULONG)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    ULONG&          nMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
nMedian	ULONG&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.29 MedianElementElementwise (USHORT)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    USHORT&         nMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
nMedian	USHORT&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.30 MedianElementElementwise (ULONGLONG)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    ULONGLONG&      nMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
nMedian	ULONGLONG&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.31 MedianElementElementwise (unsigned char)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    unsigned char&  nMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
nMedian	unsigned char&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.2.32 MedianElementElementwise (TcVnVector4_REAL)

Gets the element wise median container element.

Syntax

Definition:

```
HRESULT MedianElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector4_REAL& aMedian
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMedian	TcVnVector4_REAL [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3 MinElementElementwise

Dieses Kapitel enthält Funktionen, mit denen der Min-Wert entsprechend dem Datentyp Elementweise ermittelt wird. Diese Funktion ist für Container mit Grundelementen verfügbar.

Z. B. bei einem TcVnPoint2_DINT werden der minimale X-Wert und der minimale Y-Wert unabhängig voneinander ermittelt, daher müssen die Werte nicht von dem gleichen Punkt stammen.

6.2.3.8.3.1 MinElementElementwise (LONG)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    LONG&           nMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
nMin	LONG&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.2 MinElementElementwise (SHORT)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    SHORT&          nMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
nMin	SHORT&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.3 MinElementElementwise (double)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    double&         fMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
fMin	double&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.4 MinElementElementwise (float)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    float&          fMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
fMin	float&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.5 MinElementElementwise (char)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    char&           nMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
nMin	char&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.6 MinElementElementwise (TcVnPoint2_DINT)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnPoint2_DINT& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnPoint2_DINT [▶ 1711]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.7 MinElementElementwise (TcVnPoint2_LREAL)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnPoint2_LREAL& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnPoint2_LREAL [▶ 1711]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.8 MinElementElementwise (TcVnPoint2_REAL)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnPoint2_REAL& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnPoint2_REAL [▶ 1711]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.9 MinElementElementwise (TcVnPoint3_LREAL)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnPoint3_LREAL& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnPoint3_LREAL [▶ 1711]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.10 MinElementElementwise (TcVnPoint3_REAL)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnPoint3_REAL& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnPoint3_REAL [▶ 1711]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.11 MinElementElementwise (TcVnVector2_DINT)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_DINT& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnVector2_DINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.12 MinElementElementwise (TcVnVector2_INT)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnVector2_INT& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnVector2_INT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.13 MinElementElementwise (TcVnVector2_REAL)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_REAL& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnVector2_REAL [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.14 MinElementElementwise (TcVnVector2_SINT)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_SINT& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnVector2_SINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.15 MinElementElementwise (TcVnVector2_UINT)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector2_UINT& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnVector2_UINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.16 MinElementElementwise (TcVnVector2_USINT)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContainer,
    TcVnVector2_USINT& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnVector2_USINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.17 MinElementElementwise (TcVnVector3_INT)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnVector3_INT& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnVector3_INT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.18 MinElementElementwise (TcVnVector3_REAL)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector3_REAL& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnVector3_REAL [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.19 MinElementElementwise (TcVnVector3_SINT)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector3_SINT& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnVector3_SINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.20 MinElementElementwise (TcVnVector3_UINT)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector3_UINT& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnVector3_UINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.21 MinElementElementwise (TcVnVector3_USINT)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContainer,
    TcVnVector3_USINT& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnVector3_USINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.22 MinElementElementwise (TcVnVector4_DINT)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector4_DINT& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnVector4_DINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.23 MinElementElementwise (TcVnVector4_INT)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContainer,
    TcVnVector4_INT& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnVector4_INT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.24 MinElementElementwise (TcVnVector4_LREAL)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContainer,
    TcVnVector4_LREAL& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnVector4_LREAL [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.25 MinElementElementwise (TcVnVector4_SINT)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector4_SINT& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnVector4_SINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.26 MinElementElementwise (TcVnVector4_UINT)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector4_UINT& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnVector4_UINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.27 MinElementElementwise (TcVnVector4_USINT)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContainer,
    TcVnVector4_USINT& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnVector4_USINT [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.28 MinElementElementwise (ULONG)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    ULONG&          nMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
nMin	ULONG&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.29 MinElementElementwise (USHORT)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    USHORT&         nMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
nMin	USHORT&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.30 MinElementElementwise (ULONGLONG)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    ULONGLONG&      nMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
nMin	ULONGLONG&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.31 MinElementElementwise (unsigned char)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT         hrPrev,
    ITcVnContainer* ipContainer,
    unsigned char&  nMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
nMin	unsigned char&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.3.32 MinElementElementwise (TcVnVector4_REAL)

Gets the element wise minimum container element.

Syntax

Definition:

```
HRESULT MinElementElementwise(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContainer,
    TcVnVector4_REAL& aMin
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer	ITcVnContainer* [▶ 1890]	Source container
aMin	TcVnVector4_REAL [▶ 1713]&	Returns the requested element

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.4 ContainerAverage

Calculate the average of elements in a container.

Syntax

Definition:

```
HRESULT ContainerAverage(
    HRESULT         hrPrev,
    ITcVnContainer* ipSrcContainer,
    double&         fAverage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1890]	Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)
fAverage	double&	Returns the calculated average value

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.5 ContainerAverageElementwise (2)

Calculate the elementwise average of 2D elements in a container.

Syntax

Definition:

```
HRESULT ContainerAverageElementwise(
    HRESULT            hrPrev,
    ITcVnContainer*    ipSrcContainer,
    TcVnVector2_LREAL& aAverage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1890]	Source container with 2D points or vectors
aAverage	TcVnVector2_LREAL [▶ 1713]&	Returns the calculated average values

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.6 ContainerAverageElementwise (3)

Calculate the elementwise average of 3D elements in a container.

Syntax

Definition:

```
HRESULT ContainerAverageElementwise(
    HRESULT            hrPrev,
    ITcVnContainer*    ipSrcContainer,
    TcVnVector3_LREAL& aAverage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1890]	Source container with 3D points or vectors
aAverage	TcVnVector3_LREAL [▶ 1713]&	Returns the calculated average values

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.7 ContainerAverageElementwise (4)

Calculate the elementwise average of 4D elements in a container.

Syntax

Definition:

```
HRESULT ContainerAverageElementwise(
    HRESULT            hrPrev,
    ITcVnContainer*    ipSrcContainer,
    TcVnVector4_LREAL& aAverage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1890]	Source container with 4D vectors
aAverage	TcVnVector4_LREAL [▶ 1713]&	Returns the calculated average values

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.8 ContainerAverageVariance

Calculate the average and variance of elements in a container.

Syntax

Definition:

```
HRESULT ContainerAverageVariance(
    HRESULT         hrPrev,
    ITcVnContainer* ipSrcContainer,
    double&         fAverage,
    double&         fVariance
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1890]	Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)
fAverage	double&	Returns the calculated average value
fVariance	double&	Returns the calculated variance value

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.9 ContainerAverageVarianceElementwise (2)

Calculate the elementwise average and variance of 2D elements in a container.

Syntax

Definition:

```
HRESULT ContainerAverageVarianceElementwise(
    HRESULT            hrPrev,
    ITcVnContainer*    ipSrcContainer,
    TcVnVector2_LREAL& aAverage,
    TcVnVector2_LREAL& aVariance
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1890]	Source container with 2D points or vectors
aAverage	TcVnVector2_LREAL [▶ 1713]&	Returns the calculated average values
aVariance	TcVnVector2_LREAL [▶ 1713]&	Returns the calculated variance values

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.10 ContainerAverageVarianceElementwise (3)

Calculate the elementwise average and variance of 3D elements in a container.

Syntax

Definition:

```
HRESULT ContainerAverageVarianceElementwise(
    HRESULT            hrPrev,
    ITcVnContainer*    ipSrcContainer,
    TcVnVector3_LREAL& aAverage,
    TcVnVector3_LREAL& aVariance
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1890]	Source container with 3D points or vectors
aAverage	TcVnVector3_LREAL [▶ 1713]&	Returns the calculated average values
aVariance	TcVnVector3_LREAL [▶ 1713]&	Returns the calculated variance values

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.11 ContainerAverageVarianceElementwise (4)

Calculate the elementwise average and variance of 4D elements in a container.

Syntax

Definition:

```
HRESULT ContainerAverageVarianceElementwise(
    HRESULT            hrPrev,
    ITcVnContainer*    ipSrcContainer,
    TcVnVector4_LREAL& aAverage,
    TcVnVector4_LREAL& aVariance
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1890]	Source container with 4D vectors
aAverage	TcVnVector4_LREAL [▶ 1713]&	Returns the calculated average values
aVariance	TcVnVector4_LREAL [▶ 1713]&	Returns the calculated variance values

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.12 MaxElement

Gets the maximum element of a container.

Syntax

Definition:

```
HRESULT MaxElement(
    HRESULT                      hrPrev,
    ITcVnContainer*              ipSrcContainer,
    ITcVnBidirectionalIterator*& ipDestIterator,
    ULONGLONG&                   nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1890]	Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)
ipDestIterator	ITcVnBidirectionalIterator* [▶ 1880]&	Returns an iterator to the requested element
nIndex	ULONGLONG&	Returns the position of the requested element within the container

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.13 MedianElement

Gets the median element of a container.

Syntax

Definition:

```
HRESULT MedianElement(
    HRESULT                     hrPrev,
    ITcVnContainer*             ipSrcContainer,
    ITcVnBidirectionalIterator*& ipDestIterator,
    ULONGLONG&                  nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1890]	Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)
ipDestIterator	ITcVnBidirectionalIterator* [▶ 1880]&	Returns an iterator to the requested element
nIndex	ULONGLONG&	Returns the position of the requested element within the container

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.8.14 MinElement

Gets the minimum element of a container.

Syntax

Definition:

```
HRESULT MinElement(
    HRESULT                       hrPrev,
    ITcVnContainer*               ipSrcContainer,
    ITcVnBidirectionalIterator*&  ipDestIterator,
    ULONGLONG&                    nIndex
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContainer	ITcVnContainer* [▶ 1890]	Source container (Arithmetic, point and vector element types are supported. For points and vectors, the euclidean norm is used.)
ipDestIterator	ITcVnBidirectionalIterator* [▶ 1880]&	Returns an iterator to the requested element
nIndex	ULONGLONG&	Returns the position of the requested element within the container

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9 Contour Analysis

Die Gruppe enthält Funktionen zur Analyse von Konturen:

Funktionen

Formen approximieren

- ApproximatePolygon [▶ 2349]
- ConvexHullPoints [▶ 2362]
- EnclosingCircle [▶ 2366]
- EnclosingRectangle [▶ 2367]
- EnclosingTriangle [▶ 2367]

- FitEllipse [▶ 2368]
- FitLine [▶ 2369]
- UprightBoundingRectangle [▶ 2376]

Punkte

- CheckIfPointIsInsideContour [▶ 2350]
- ContourExtremePoint [▶ 2357]
- ConvexityDefects [▶ 2363]

Geometrische Merkmale

- ContourArea [▶ 2351]
- ContourCenterOfMass [▶ 2352]
- ContourCircularity [▶ 2353]
- ContourConvexity [▶ 2354]
- ContourEccentricity [▶ 2355]
- ContourElongation [▶ 2356]
- ContourInertiaRatio [▶ 2357]
- ContourMoments [▶ 2358]
- ContourOrientation [▶ 2359]
- ContourPerimeter [▶ 2360]
- ContourRoundness [▶ 2361]
- FourierDescriptors [▶ 2371]

Matching

- MatchContours [▶ 2372]
- MatchContours1vsN [▶ 2374]

Benutzerdefinierte Operationen

- CustomElementWiseContainerOperation_ITcVnContainer [▶ 2364]
- CustomElementWiseContainerOperation_ITcVnForwardIterator [▶ 2365]

6.2.3.9.1 ApproximatePolygon

Approximate a contour to a simplified polygon (using the Douglas-Peucker algorithm).

Syntax

Definition:

```
HRESULT ApproximatePolygon(
    HRESULT          hrPrev,
    ITcVnContainer*  ipSrcContour,
    ITcVnContainer*& ipDestContour,
    double           fMaxDist,
    bool             bClosed
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcContour	ITcVnContainer* [▶ 1890]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
ipDestContour	ITcVnContainer* [▶ 1890]&	Returns a container which is filled with the approximated polygon points (same type ID as ipSrcContour; Non-zero interface pointers are reused.)
fMaxDist	double	Maximum distance between the original contour and its approximation
bClosed	bool	Specify, if the contour is closed (first and last points connected) or not

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.2 CheckIfPointIsInsideContour

Checks if a point is inside a contour (and optionally return the distance to it).

Syntax

Definition:

```
HRESULT CheckIfPointIsInsideContour(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContour,
    TcVnPoint2_REAL& aPoint,
    double&          fDist,
    bool             bMeasureDistance
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1890]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL)
aPoint	TcVnPoint2_REAL [▶ 1711]&	Point position to check
fDist	double&	Returns the result (> 0: the point is inside the contour; 0: the point is on the contour; < 0: the point is outside the contour)
bMeasureDistance	bool	If true, fDist returns the distance to the nearest contour edge. Otherwise, fDist only returns -1, 0 or +1 (recommended to set to false if the distance is not required, as this is faster).

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.3 ContourArea

Estimate the contour area using Green's formula.

Syntax

Definition:

```
HRESULT ContourArea(
    HRESULT         hrPrev,
    ITcVnContainer* ipContour,
    double&         fArea
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1890]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
fArea	double&	Returns the estimated contour area (The actual contour area may differ depending on the shape of the contour.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.4 ContourCenterOfMass

Computes the center of mass of a contour.

Syntax

Definition:

```
HRESULT ContourCenterOfMass(
    HRESULT           hrPrev,
    ITcVnContainer*   ipContour,
    TcVnPoint2_LREAL& aCenterOfMass
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1890]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
aCenterOfMass	TcVnPoint2_LREAL [▶ 1711]&	Returns the center of mass of the contour

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.5 ContourCircularity

Computes the circularity of a contour (1.0: ideal circle, 0.0: straight line).

Syntax

Definition:

```
HRESULT ContourCircularity(
    HRESULT                          hrPrev,
    ITcVnContainer*                  ipContour,
    double&                          fCircularity,
    ETcVnCircularityCalculationMethod eMethod = CCM_DEFAULT
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1890]		Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
fCircularity	double&		Returns the circularity of the contour [0..1]
eMethod	ETcVnCircularityCalculationMethod [▶ 1761]	CCM_DEFAULT	Method used for computing the circularity

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.6 ContourConvexity

Computes the convexity of a contour as the ratio between the contour's area and the area of the convex hull (1.0: convex shapes (i.e. rectangle, circle, ..., etc.), 0.0: zero area (i.e. a line or an empty contour)).

Syntax

Definition:

```
HRESULT ContourConvexity(
    HRESULT         hrPrev,
    ITcVnContainer* ipContour,
    double&         fConvexity
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1890]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
fConvexity	double&	Returns the convexity of the contour [0..1]

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.7 ContourEccentricity

Computes the eccentricity of a contour (0.0: circular, 1.0: linear).

Syntax

Definition:

```
HRESULT ContourEccentricity(
    HRESULT         hrPrev,
    ITcVnContainer* ipContour,
    double&         fEccentricity
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1890]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
fEccentricity	double&	Returns the eccentricity of the contour [0..1]

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.8 ContourElongation

Computes the elongation factor of a contour.

Syntax

Definition:

```
HRESULT ContourElongation(
    HRESULT         hrPrev,
    ITcVnContainer* ipContour,
    double&         fElongation
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1890]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
fElongation	double&	Returns the elongation of the contour

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.9 ContourExtremePoint

Finds the contour extreme point.

Syntax

Definition:

```
HRESULT ContourExtremePoint(
    HRESULT                    hrPrev,
    ITcVnContainer*            ipContour,
    ETcVnExtremePointDirection eDirection,
    TcVnPoint2_LREAL&          aExtremePoint
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1890]	Source container with 2D points (TcVnPoint2_DINT or TcVnPoint2_REAL or TcVnPoint2_LREAL)
eDirection	ETcVnExtremePointDirection [▶ 1739]	Selects the search direction for the extreme point
aExtremePoint	TcVnPoint2_LREAL [▶ 1711]&	Returns the extreme point of the contour, according to eDirection

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.10 ContourInertiaRatio

Computes the inertia ratio of a contour which reflects how a contour/shape is elongated (i.e. circle: 1.0, line: 0.0).

Syntax

Definition:

```
HRESULT ContourInertiaRatio(
    HRESULT         hrPrev,
    ITcVnContainer* ipContour,
    double&         fInertiaRatio
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1890]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
fInertiaRatio	double&	Returns the inertia raio of the contour [0..1]

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.11 ContourMoments

Computes the spatial moments, the central moments, and the central normalized moments of a contour up to the third order.

Syntax

Definition:

```
HRESULT ContourMoments(
    HRESULT         hrPrev,
    ITcVnContainer* ipContour,
    TcVnMoments&    stMoments
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1890]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
stMoments	TcVnMoments [▶ 1769]&	Returns a struct containing the moments

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.12 ContourOrientation

Calculate the orientation of a contour or a set of points.

Syntax

Definition:

```
HRESULT ContourOrientation(
    HRESULT                hrPrev,
    ITcVnContainer*        ipContour,
    TcVnRotatedRectangle&  stOrientation,
    ETcVnOrientationMethod eMethod = OM_PCA
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1890]		Source point set (ContainerType_Vector_TcVnPoint2_DINT, ContainerType_Vector_TcVnPoint2_REAL, or ContainerType_Vector_TcVnPoint2_LREAL. The elements of this container are the individual points.)
stOrientation	TcVnRotatedRectangle [▶ 1782]&		Resulting rotated rectangle, containig center of mass, lenghts of axes, and rotation angle of the contour in clockwise direction.
eMethod	ETcVnOrientationMethod [▶ 1749]	OM_PCA	Method for calculating the orientation.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.13 ContourPerimeter

Computes the perimeter of a contour (curve length if the contour is not closed).

Syntax

Definition:

```
HRESULT ContourPerimeter(
    HRESULT         hrPrev,
    ITcVnContainer* ipContour,
    double&         fPerimeter,
    bool            bClosed
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1890]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
fPerimeter	double&	Returns the perimeter of the contour
bClosed	bool	Specifys, if the contour is closed (first and last points connected) or not

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.14 ContourRoundness

Computes the roundness of a contour (perimeter^2 / area).

Syntax

Definition:

```
HRESULT ContourRoundness(
    HRESULT         hrPrev,
    ITcVnContainer* ipContour,
    double&         fRoundness
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1890]	Single contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour.)
fRoundness	double&	Returns the roundness of the contour

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.15 ConvexHullPoints

Determines the convex hull of a point set.

Syntax

Definition:

```
HRESULT ConvexHullPoints(
    HRESULT          hrPrev,
    ITcVnContainer*  ipPointSet,
    ITcVnContainer*& ipConvexHull,
    bool             bClockwise = false
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipPointSet	ITcVnContainer* [▶ 1890]		Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
ipConvexHull	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the sorted convex hull points (same type ID as ipPointSet; Non-zero interface pointers are reused.)
bClockwise	bool	false	Selects the sorting direction of the hull points, assuming cartesian coordinates.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.16 ConvexityDefects

Finds the convexity defects of a point set.

Syntax

Definition:

```
HRESULT ConvexityDefects(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContour,
    ITcVnContainer*& ipConvexityDefects
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1890]	Source contour (ContainerType_Vector_TcVnPoint2_DINT; The elements of this container are the individual points.)
ipConvexityDefects	ITcVnContainer* [▶ 1890]&	Returns a container which is filled with the convexity defects (ContainerType_Vector_TcVnVector4_DINT with [startIndex, endIndex, furthestPointIndex, fixedPointDistance (8 fraction bits, i.e. divide by 256 to get the distance in pixels)]; Non-zero interface pointers are reused.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.17 CustomElementWiseContainerOperation (ITcVnContainer*)

Performs a custom, element wise operation on a set of up to 3 containers.

Syntax

Definition:

```
HRESULT CustomElementWiseContainerOperation(
    HRESULT                                     hrPrev,
    ITcVnContainer*                             ipContainer1,
    ITcVnContainer*                             ipContainer2,
    ITcVnContainer*                             ipContainer3,
    ITcVnCustomContainerOperation_ITcVnContainer* ipOperationFB
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer1	ITcVnContainer* [▶ 1890]	Container 1
ipContainer2	ITcVnContainer* [▶ 1890]	Container 2
ipContainer3	ITcVnContainer* [▶ 1890]	Container 3
ipOperationFB	ITcVnCustomContainerOperation_ITcVnContainer* [▶ 1783]	Custom operation on the container elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.18 CustomElementWiseContainerOperation (ITcVnForwardIterator*)

Performs a custom, element wise operation on a set of up to 3 containers.

Syntax

Definition:

```
HRESULT CustomElementWiseContainerOperation(
    HRESULT                                          hrPrev,
    ITcVnContainer*                                  ipContainer1,
    ITcVnContainer*                                  ipContainer2,
    ITcVnContainer*                                  ipContainer3,
    ITcVnCustomContainerOperation_ITcVnForwardIterator* ipOperationFB
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContainer1	ITcVnContainer* [▶ 1890]	Container 1
ipContainer2	ITcVnContainer* [▶ 1890]	Container 2
ipContainer3	ITcVnContainer* [▶ 1890]	Container 3
ipOperationFB	ITcVnCustomContainerOperation_ITcVnForwardIterator* [▶ 1784]	Custom operation on the elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.19 EnclosingCircle

Searchs for a minimum area circle enclosing a set of points.

Syntax

Definition:

```
HRESULT EnclosingCircle(
    HRESULT          hrPrev,
    ITcVnContainer*  ipPointSet,
    TcVnPoint2_REAL& aCenter,
    float&           fRadius
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipPointSet	ITcVnContainer* [▶ 1890]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
aCenter	TcVnPoint2_REAL [▶ 1711]&	Returns the center of the circle
fRadius	float&	Returns the radius of the circle

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.20 EnclosingRectangle

Searches for a minimum area rectangle enclosing a set of points.

Syntax

Definition:

```
HRESULT EnclosingRectangle(
    HRESULT               hrPrev,
    ITcVnContainer*       ipPointSet,
    TcVnRotatedRectangle& stRectangle
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipPointSet	ITcVnContainer* [▶ 1890]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
stRectangle	TcVnRotatedRectangle [▶ 1782]&	Returns the determined rectangle

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.21 EnclosingTriangle

Searches for a minimum area triangle enclosing a set of points.

Syntax

Definition:

```
HRESULT EnclosingTriangle(
    HRESULT                 hrPrev,
    ITcVnContainer*         ipPointSet,
    TcVnArray3_Point2_REAL& aTriangleVertices,
    double&                 fArea
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipPointSet	ITcVnContainer* [▶ 1890]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
aTriangleVertices	TcVnArray3_Point2_REAL [▶ 1713]&	Returns the 3 triangle vertices
fArea	double&	Returns the triangle area

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.22 FitEllipse

Fit an ellipse in a point set.

Syntax

Definition:

```
HRESULT FitEllipse(
    HRESULT               hrPrev,
    ITcVnContainer*       ipPointSet,
    TcVnRotatedRectangle& stEllipse
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipPointSet	ITcVnContainer* [▶ 1890]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; At least 5 reasonable points are required! The elements of this container are the individual points.)
stEllipse	TcVnRotatedRectangle [▶ 1782]&	Resulting ellipse, described by a rotated rectangle.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.23 FitLine

Fit a line into a point set.

Syntax

Definition:

```
HRESULT FitLine(
    HRESULT            hrPrev,
    ITcVnContainer*    ipPointSet,
    TcVnVector4_LREAL& aFitLine,
    ETcVnDistanceType  eDistanceType = DT_L2,
    double             fParam = 0,
    double             fEpsRadius = 0.001,
    double             fEpsAngle = 0.001
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipPointSet	ITcVnContainer* [▶ 1890]		Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
aFitLine	TcVnVector4_LREAL [▶ 1713]&		Resulting line. The first and second element describe the x and y component of a vector collinear to the line. The third and fourth element describe the x and y component of a point on the line.
eDistanceType	ETcVnDistanceType [▶ 1736]	DT_L2	Distance computation method (supported: L1, L2, L12, FAIR, WELSCH, HUBER)
fParam	double	0	Numerical parameter (c) for some eDistanceType (should be >= 0). If 0, an optimal value is chosen internally.
fEpsRadius	double	0.001	Accuracy of the radius (distance of the line from the coordinate origin, should be > 0). A smaller value means higher accuracy.
fEpsAngle	double	0.001	Accuracy of the angle (should be > 0). A smaller value means higher accuracy.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.24 FourierDescriptors

Computes the fourier descriptors for a closed contour.

Syntax

Definition:

```
HRESULT FourierDescriptors(
    HRESULT          hrPrev,
    ITcVnContainer*  ipContour,
    ITcVnContainer*& ipDescriptors,
    LONG             nDescriptors
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour	ITcVnContainer* [▶ 1890]	Closed contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL. Providing the full contour is strongly recommended, i.e. use TCVN_CAM_NONE for contour detection algorithms.)
ipDescriptors	ITcVnContainer* [▶ 1890]&	Returns the fourier descriptors for ipContour (Re0, Im0, Re1, Im1, ...; ContainerType_Vector_LREAL if ipContour is of type ContainerType_Vector_TcVnPoint2_LREAL, else ContainerType_Vector_REAL.)
nDescriptors	LONG	Specifies how many descriptors should be returned (set to -1 to return all computed descriptors, i.e. at least as many as the number of contour points)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.25 MatchContours

Compare contours using the Hu moment invariants (and optionally further aspects). In case of multiple contours in each container, the best matches are found and the average dissimilarity over all matched contours is returned.

Syntax

Definition:

```
HRESULT MatchContours(
    HRESULT                              hrPrev,
    ITcVnContainer*                      ipContour1,
    ITcVnContainer*                      ipContour2,
    ETcVnContoursMatchComparisonMethod eComparisonMethod,
    double&                              fDissimilarity,
    double                               fAreaFactor = 0,
    double                               fAbsPositionFactor = 0,
    double                               fRelPositionFactor = 0,
    double                               fNumDiffFactor = 0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContour1	ITcVnContainer* [▶ 1890]		First contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour) or collection of multiple contours (ContainerType_Vector_Vector_TcVnPoint2_DINT)
ipContour2	ITcVnContainer* [▶ 1890]		Second contour (same type as ipContour1)
eComparisonMethod	ETcVnContoursMatchComparisonMethod [▶ 1734]		Method used for comparing the Hu moment invariants of the contours
fDissimilarity	double&		Returns the dissimilarity of the contours depending on the chosen comparison method
fAreaFactor	double	0	If > 0, the relative area difference between contours is scaled by this factor and added to the computed dissimilarity
fAbsPositionFactor	double	0	If > 0, the absolute position difference between contours (i.e. the coordinates of the geometric contour centers) is scaled by this factor and added to the computed dissimilarity
fRelPositionFactor	double	0	If > 0, the relative position difference between contours is scaled by this factor and added to the computed dissimilarity (only used if matching multiple contours)
fNumDiffFactor	double	0	If > 0, the difference between the number of contours in both containers is scaled by this factor and added to the computed dissimilarity (only used if matching multiple contours)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.26 MatchContours1vsN

Compare a reference contour with multiple other contours using the Hu moment invariants (and optionally further aspects). Returns a sorted list of best matches.

Syntax

Definition:

```
HRESULT MatchContours1vsN(
    HRESULT                             hrPrev,
    ITcVnContainer*                     ipRefContour,
    ITcVnContainer*                     ipContours,
    ITcVnContainer*&                    ipMatchIndexes,
    ITcVnContainer*&                    ipDissimilarities,
    double                              fDissimilarityThreshold,
    ETcVnContoursMatchComparisonMethod eComparisonMethod,
    double                              fAreaFactor = 0,
    double                              fAbsPositionFactor = 0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipRefContour	ITcVnContainer* [▶ 1890]		Reference contour (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the points of the contour)
ipContours	ITcVnContainer* [▶ 1890]		Collection of multiple contours (CTcVnContainer_Vector_Vector_TcVnPoint2_DINT)
ipMatchIndexes	ITcVnContainer* [▶ 1890]&		Returns the indexes of the best matches (CTcVnContainer_Vector_ULINT; sorted, first element is best match, i.e. has lowest dissimilarity)
ipDissimilarities	ITcVnContainer* [▶ 1890]&		Returns the computed dissimilarities of the best matches (CTcVnContainer_Vector_LREAL; Optional, set to 0 if not required; sorted corresponding to ipMatchIndexes)
fDissimilarityThreshold	double		Neglect irrelevant matches, i.e. dissimilarity > fDissimilarityThreshold
eComparisonMethod	ETcVnContoursMatchComparisonMethod [▶ 1734]		Method used for comparing the Hu moment invariants of the contours
fAreaFactor	double	0	If > 0, the relative area difference between contours is scaled by this factor and added to the computed dissimilarity
fAbsPositionFactor	double	0	If > 0, the absolute position difference between contours (i.e. the coordinates of the geometric contour centers) is scaled by this factor and added to the computed dissimilarity

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.9.27 UprightBoundingRectangle

Determines the upright bounding rectangle of a set of points.

Syntax

Definition:

```
HRESULT UprightBoundingRectangle(
    HRESULT              hrPrev,
    ITcVnContainer*      ipPointSet,
    TcVnRectangle_UDINT& stRectangle
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipPointSet	ITcVnContainer* [▶ 1890]	Source point set (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL; The elements of this container are the individual points.)
stRectangle	TcVnRectangle_UDINT [▶ 1782]&	Returns the determined rectangle

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.10 Control Functions

Diese Gruppe enthält Funktionen zur Ablaufsteuerung, z. B. zur Handhabung von Watchdogs [▶ 156].

Funktionen

Watchdogs

- StartAbsWatchdog [▶ 2377]
- StartRelWatchdog [▶ 2377]
- StopWatchdog [▶ 2378]

6.2.3.10.1 StartAbsWatchdog

Start a cooperative watchdog given an absolute stop time.

Syntax

Definition:

```
HRESULT StartAbsWatchdog(
    HRESULT                     hrPrev,
    LONG                        tStop,
    ETcWatchdogAccumulationType eWatchdogAccType = WD_ACC_TYPE_PRODUCT
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
tStop	LONG		Stop time in us
eWatchdogAccType	ETcWatchdogAccumulationType [▶ 1759]	WD_ACC_TYPE_PRODUCT	Accumalation method used for combining the results of multiple functions enclosed by this watchdog

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.10.2 StartRelWatchdog

Start a cooperative watchdog given a stop time relative to the current time.

Syntax

Definition:

```
HRESULT StartRelWatchdog(
    HRESULT                     hrPrev,
    LONG                        tStop,
    ETcWatchdogAccumulationType eWatchdogAccType = WD_ACC_TYPE_PRODUCT
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
tStop	LONG		Stop time in us
eWatchdogAccType	ETcWatchdogAccumulationType [▶ 1759]	WD_ACC_TYPE_PRODUCT	Accumalation method used for combining the results of multiple functions enclosed by this watchdog

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.10.3 StopWatchdog

Stops a watchdog and provide runtime information.

Syntax

Definition:

```
HRESULT StopWatchdog(
    HRESULT   hrStartWatchdog,
    ULONGLONG& nFunctionsMonitored,
    ULONG&    nFractionProcessed,
    LONG&     tRest
)
```

Parameters

Name	Type	Description
hrStartWatchdog	HRESULT [▶ 150]	HRESULT indicating the result of the function used to start the watchdog (If SUCCEEDED(hrStartWatchdog) equals false, no operation is executed.)
nFunctionsMonitored	ULONGLONG&	Returns the number of functions monitored
nFractionProcessed	ULONG&	Returns the fraction processed accumulated over the monitored functions in percent
tRest	LONG&	Returns the remaining computation time in us (may be negative)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11 Drawing

Diese Gruppe enthält Funktionen zum Bemalen und Beschreiben von Bildern.

Handhabung von 4-kanaligen Farben

4-kanalige Farben werden direkt in das Ergebnisbild übernommen. Es wird kein Alpha-Blending o. Ä. betrieben.

Funktionen

Arrow

- DrawArrow [▶ 2380]
- DrawArrow_TcVnVector4_DINT [▶ 2381]

Circles und Arcs

- DrawCircle [▶ 2382]
- DrawCircles [▶ 2383]
- DrawCircularArc [▶ 2384]
- FillCircle [▶ 2404]

Components

- DrawComponents [▶ 2385]

Contours

- DrawContours [▶ 2386]
- FillContours [▶ 2404]

Ellipse

- DrawEllipse [▶ 2387]
- FillEllipse [▶ 2405]

Keypoints und Matches

- DrawKeypoints [▶ 2388]
- DrawMatches [▶ 2394]

Lines

- DrawLine [▶ 2389]
- DrawLines [▶ 2392]
- DrawLine_TcVnVector4_DINT [▶ 2390]

- DrawLine_TcVnVector4_LREAL [▶ 2391]

Orientation

- DrawOrientation(Exp) [▶ 2396]

Points

- DrawPoint [▶ 2397]
- DrawPoints [▶ 2398]

Polygon

- DrawPolygon [▶ 2399]
- FillPolygon [▶ 2406]

Rectangle

- DrawRectangle [▶ 2400]
- DrawRectangle_TcVnRectangle_DINT [▶ 2401]
- DrawRectangle_TcVnRectangle_UDINT [▶ 2402]
- DrawRotatedRectangle [▶ 2403]
- FillRectangle [▶ 2407]
- FillRotatedRectangle [▶ 2408]

Intensity Profile

- PlotIntensityProfile [▶ 2408]

Text

- PutLabel [▶ 2410]
- PutText [▶ 2411]

Malen und Füllen

Für viele Formen gibt es die Möglichkeit, sowohl den Umriss der Form als auch die gefüllte Form zu malen. Dazu stehen je zwei Funktionsgruppen zur Verfügung (Draw… und Fill…). Außerdem führt ein Setzen der Umrissdicke nThickness auf -1 auch dazu, dass die Form gefüllt wird.

6.2.3.11.1 DrawArrow

Draws an arrowed line.

Syntax

Definition:

```
HRESULT DrawArrow(
    HRESULT             hrPrev,
    ULONG               nX1,
    ULONG               nY1,
    ULONG               nX2,
    ULONG               nY2,
    ITcVnImage*         ipDestImage,
    TcVnVector4_LREAL&  aColor,
    LONG                nThickness,
    bool                bDoubleHead,
    ETcVnLineType       eLineType = LT_8_CONNECTED,
    ULONG               nShift = 0,
    double              fTipLenght = 0.1
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nX1	ULONG		x coordinate of the start point
nY1	ULONG		y coordinate of the start point
nX2	ULONG		x coordinate of the end point
nY2	ULONG		y coordinate of the end point
ipDestImage	ITcVnImage* [▶ 1925]		Destination image
aColor	TcVnVector4_LREAL [▶ 1713]&		Color
nThickness	LONG		Line thickness
bDoubleHead	bool		To draw an arrow on each side of the line.
eLineType	ETcVnLineType [▶ 1745]	LT_8_CONNECTED	Line type
nShift	ULONG	0	Fractional bits of the coordinates (bit shift)
fTipLenght	double	0.1	Lenght of the arrow's tip as a fraction of the line's lenght

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.2 DrawArrow (TcVnVector4_DINT)

Draws an arrowed line.

Syntax

Definition:

```
HRESULT DrawArrow(
    HRESULT            hrPrev,
    TcVnVector4_DINT&  aLine,
    ITcVnImage*        ipDestImage,
    TcVnVector4_LREAL& aColor,
    LONG               nThickness,
    bool               bDoubleHead,
```

```
    ETcVnLineType       eLineType = LT_8_CONNECTED,
    ULONG               nShift = 0,
    double              fTipLenght = 0.1
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aLine	TcVnVector4_DINT [▶ 1713]&		The start and end point of a line segment [x1, y1, x2, y2]
ipDestImage	ITcVnImage* [▶ 1925]		Destination image
aColor	TcVnVector4_LREAL [▶ 1713]&		Color
nThickness	LONG		Line thickness
bDoubleHead	bool		To draw an arrow on each side of the line.
eLineType	ETcVnLineType [▶ 1745]	LT_8_CONNECTED	Line type
nShift	ULONG	0	Fractional bits of the coordinates (bit shift)
fTipLenght	double	0.1	Lenght of the arrow's tip as a fraction of the line's lenght

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.3 DrawCircle

Draws a circle.

Syntax

Definition:

```
HRESULT DrawCircle(
    HRESULT             hrPrev,
    ULONG               nCenterX,
    ULONG               nCenterY,
    ULONG               nRadius,
    ITcVnImage*         ipDestImage,
    TcVnVector4_LREAL&  aColor,
```

```
    LONG                nThickness,
    ETcVnLineType       eLineType = LT_8_CONNECTED
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nCenterX	ULONG		x coordinate of the center
nCenterY	ULONG		y coordinate of the center
nRadius	ULONG		Radius
ipDestImage	ITcVnImage* [▶ 1925]		Destination image
aColor	TcVnVector4_LREAL [▶ 1713]&		Color
nThickness	LONG		Line thickness (if negative, the circle is filled)
eLineType	ETcVnLineType [▶ 1745]	LT_8_CONNECTED	Line type

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.4 DrawCircles

Draws circles.

Syntax

Definition:

```
HRESULT DrawCircles(
    HRESULT             hrPrev,
    ITcVnContainer*     ipCircles,
    LONG                nCircleIndex,
    ITcVnImage*         ipDestImage,
    TcVnVector4_LREAL&  aColor,
    LONG                nThickness,
    ETcVnLineType       eLineType = LT_8_CONNECTED
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipCircles	ITcVnContainer* [▶ 1890]		Container with circles (ContainerType_Vector_TcVnVector3_REAL; Each container element contains the x coordinate of the circle center [0], the y coordinate of the circle center [1], and the radius [2].)
nCircleIndex	LONG		Index of a specific circle to be drawn (if negative, all circles within the container are drawn)
ipDestImage	ITcVnImage* [▶ 1925]		Destination image
aColor	TcVnVector4_LREAL [▶ 1713]&		Color
nThickness	LONG		Line thickness (if negative, the circle is filled)
eLineType	ETcVnLineType [▶ 1745]	LT_8_CONNECTED	Line type

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.5 DrawCircularArc

Draws a circular arc.

Syntax

Definition:

```
HRESULT DrawCircularArc(
    HRESULT            hrPrev,
    TcVnCircularArc&   stCircularArc,
    ITcVnImage*        ipDestImage,
    TcVnVector4_LREAL& aColor,
    LONG               nThickness,
    ETcVnLineType      eLineType = LT_8_CONNECTED
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stCircularArc	TcVnCircularArc [▶ 1765]&		Circular arc definition
ipDestImage	ITcVnImage* [▶ 1925]		Destination image
aColor	TcVnVector4_LREAL [▶ 1713]&		Color
nThickness	LONG		Line thickness (if negative, the arc is filled)
eLineType	ETcVnLineType [▶ 1745]	LT_8_CONNECTED	Line type

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.6 DrawComponents

Draw the results of connected component function.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT DrawComponents(
    HRESULT             hrPrev,
    ITcVnImage*         ipLabelImage,
    LONG                nLabelID,
    ITcVnImage*&        ipDestImage,
    bool                bShowLabelID,
    TcVnVector4_LREAL&  aColor = {-1, -1, -1, -1},
    LONG                nThickness = 1,
    double              fFontScale = 1
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipLabelImage	ITcVnImage* [▶ 1925]		Image contains labels for each pixel (1 channel DINT or UINT).
nLabelID	LONG		Value of a specific label to be drawn (if negative, all components within the image are drawn)
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (An appropriate image will be created if required. The output image is 1 or 3 channels USINT).
bShowLabelID	bool		Show lable ID in the center of the component on the destination image
aColor	TcVnVector4_LREAL [▶ 1713]&	{-1, -1, -1, -1}	Color to draw the components (for {-1, -1, -1, -1}, a random color is chosen for each component)
nThickness	LONG	1	Text line thickness
fFontScale	double	1	Scaling factor for the text

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.7 DrawContours

Draws a single point set or multiple point sets that are interpreted as contours.

Syntax

Definition:

```
HRESULT DrawContours(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContours,
    LONG               nContourIndex,
    ITcVnImage*        ipDestImage,
    TcVnVector4_LREAL& aColor,
    LONG               nThickness,
```

```
    ETcVnLineType       eLineType = LT_8_CONNECTED,
    ITcVnContainer*     ipHierarchy = nullptr,
    LONG                nMaxLevel = 2147483647,
    TcVnPoint&          aOffset = {0, 0}
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContours	ITcVnContainer* [▶ 1890]		Single contour (ContainerType_Vector_TcVnPoint2_DINT) or multiple contours (ContainerType_Vector_Vector_TcVnPoint2_DINT)
nContourIndex	LONG		Index of a specific contour to be drawn (if negative, all contours within the container are drawn)
ipDestImage	ITcVnImage* [▶ 1925]		Destination image
aColor	TcVnVector4_LREAL [▶ 1713]&		Color
nThickness	LONG		Line thickness (if negative, the contours are filled)
eLineType	ETcVnLineType [▶ 1745]	LT_8_CONNECTED	Line type
ipHierarchy	ITcVnContainer* [▶ 1890]	nullptr	Contour hierarchy (ContainerType_Vector_TcVnVector4_DINT)
nMaxLevel	LONG	2147483647	Maximum level of contours to be drawn
aOffset	TcVnPoint [▶ 1711]&	{0, 0}	Offset by which every contour point is shifted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.8 DrawEllipse

Draws an ellipse.

Syntax

Definition:

```
HRESULT DrawEllipse(
    HRESULT                hrPrev,
    TcVnRotatedRectangle&  stEllipse,
    ITcVnImage*            ipDestImage,
    TcVnVector4_LREAL&     aColor,
    LONG                   nThickness,
    ETcVnLineType          eLineType = LT_8_CONNECTED
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stEllipse	TcVnRotatedRectangle [▶ 1782]&		Ellipse to be drawn (rotation angle in degrees)
ipDestImage	ITcVnImage* [▶ 1925]		Destination image
aColor	TcVnVector4_LREAL [▶ 1713]&		Color
nThickness	LONG		Line thickness (if negative, the ellipse is filled)
eLineType	ETcVnLineType [▶ 1745]	LT_8_CONNECTED	Line type

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.9 DrawKeypoints

Draws the keypoints for visualization purpose.

Syntax

Definition:

```
HRESULT DrawKeypoints(
    HRESULT                hrPrev,
    ITcVnImage*            ipSrcImage,
    ITcVnContainer*        ipKeyPoints,
    ITcVnImage*&           ipDestImage,
    TcVnVector4_LREAL&     aColor = {-1, -1, -1, -1},
    ETcVnDrawMatchesFlags  eFlags = DMF_DEFAULT
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (elements of type USINT)
ipKeyPoints	ITcVnContainer* [▶ 1890]		Container with the keypoints (ContainerType_Vector_TcVnKeyPoint)
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (An appropriate color image will be created if required.)
aColor	TcVnVector4_LREAL [▶ 1713]&	{-1, -1, -1, -1}	Color to draw the keypoints (for {-1, -1, -1, -1}, a random color is chosen for each point)
eFlags	ETcVnDrawMatchesFlags [▶ 1736]	DMF_DEFAULT	A combination of flags to support overdrawing an existing destination image and/or drawing additional (Rich-)Keypoint information (size and orientation)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.10 DrawLine

Draws a line.

Syntax

Definition:

```
HRESULT DrawLine(
    HRESULT            hrPrev,
    ULONG              nX1,
    ULONG              nY1,
    ULONG              nX2,
    ULONG              nY2,
    ITcVnImage*        ipDestImage,
    TcVnVector4_LREAL& aColor,
    LONG               nThickness,
```

```
    ETcVnLineType       eLineType = LT_8_CONNECTED,
    ULONG               nShift = 0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nX1	ULONG		x coordinate of the start point
nY1	ULONG		y coordinate of the start point
nX2	ULONG		x coordinate of the end point
nY2	ULONG		y coordinate of the end point
ipDestImage	ITcVnImage* [▶ 1925]		Destination image
aColor	TcVnVector4_LREAL [▶ 1713]&		Color
nThickness	LONG		Line thickness
eLineType	ETcVnLineType [▶ 1745]	LT_8_CONNECTED	Line type
nShift	ULONG	0	Fractional bits of the coordinates (bit shift)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.11 DrawLine (TcVnVector4_DINT)

Draws a line.

Syntax

Definition:

```
HRESULT DrawLine(
    HRESULT             hrPrev,
    TcVnVector4_DINT&   aLine,
    ITcVnImage*         ipDestImage,
    TcVnVector4_LREAL&  aColor,
    LONG                nThickness,
    ETcVnLineType       eLineType = LT_8_CONNECTED,
    ULONG               nShift = 0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aLine	TcVnVector4_DINT [▶ 1713]&		The start and end point of a line segment [x1, y1, x2, y2]
ipDestImage	ITcVnImage* [▶ 1925]		Destination image
aColor	TcVnVector4_LREAL [▶ 1713]&		Color
nThickness	LONG		Line thickness
eLineType	ETcVnLineType [▶ 1745]	LT_8_CONNECTED	Line type
nShift	ULONG	0	Fractional bits of the coordinates (bit shift)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.12 DrawLine (TcVnVector4_LREAL)

Draws a line.

Syntax

Definition:

```
HRESULT DrawLine(
    HRESULT            hrPrev,
    TcVnVector4_LREAL& aLine,
    ITcVnImage*        ipDestImage,
    TcVnVector4_LREAL& aColor,
    LONG               nThickness,
    ETcVnLineType      eLineType = LT_8_CONNECTED,
    ULONG              nShift = 0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aLine	TcVnVector4_LREAL [▶ 1713]&		The first and second element describe the x and y component of the direction vector. The third and fourth element describe the x and y component of the position vector.
ipDestImage	ITcVnImage* [▶ 1925]		Destination image
aColor	TcVnVector4_LREAL [▶ 1713]&		Color
nThickness	LONG		Line thickness
eLineType	ETcVnLineType [▶ 1745]	LT_8_CONNECTED	Line type
nShift	ULONG	0	Fractional bits of the coordinates (bit shift)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.13 DrawLines

Draws lines.

Syntax

Definition:

```
HRESULT DrawLines(
    HRESULT            hrPrev,
    ITcVnContainer*    ipLines,
    LONG               nLineIndex,
    ITcVnImage*        ipDestImage,
    TcVnVector4_LREAL& aColor,
    LONG               nThickness,
    ETcVnLineType      eLineType = LT_8_CONNECTED,
    ULONG              nShift = 0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipLines	ITcVnContainer* [▶ 1890]		Container with line descriptions (ContainerType_Vector_TcVnVector2_REAL: the distance from the origin [0] in pixels and the rotation angle [1] in radians. ContainerType_Vector_TcVnVector4_LREAL: the first and second element describe the x and y component of a vector collinear to the line. The third and fourth element describe the x and y component of a point on the line. ContainerType_Vector_TcVnVector4_DINT: start and end point [x1, y1, x2, y2])
nLineIndex	LONG		Index of a specific line to be drawn (if negative, all lines within the container are drawn)
ipDestImage	ITcVnImage* [▶ 1925]		Destination image
aColor	TcVnVector4_LREAL [▶ 1713]&		Color
nThickness	LONG		Line thickness
eLineType	ETcVnLineType [▶ 1745]	LT_8_CONNECTED	Line type
nShift	ULONG	0	Fractional bits of the coordinates (bit shift)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.14 DrawMatches

Draws the keypoints and matches for visualization purpose.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT DrawMatches(
    HRESULT               hrPrev,
    ITcVnImage*           ipSrcImage1,
    ITcVnContainer*       ipKeyPoints1,
    ITcVnImage*           ipSrcImage2,
    ITcVnContainer*       ipKeyPoints2,
    ITcVnContainer*       ipMatches1To2,
    ITcVnImage*&          ipDestImage,
    TcVnVector4_LREAL&    aMatchColor = {-1, -1, -1, -1},
    TcVnVector4_LREAL&    aSingleColor = {-1, -1, -1, -1},
    ITcVnContainer*       ipMatchesMask = nullptr,
    ETcVnDrawMatchesFlags eFlags = DMF_DEFAULT
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage1	ITcVnImage* [▶ 1925]		Source image 1 (elements of type USINT)
ipKeyPoints1	ITcVnContainer* [▶ 1890]		Container with the keypoints 1 (ContainerType_Vector_TcVnKeyPoint)
ipSrcImage2	ITcVnImage* [▶ 1925]		Source image 2 (elements of type USINT, must have the same number of channels as ipSrcImage1)
ipKeyPoints2	ITcVnContainer* [▶ 1890]		Container with the keypoints 2 (ContainerType_Vector_TcVnKeyPoint)
ipMatches1To2	ITcVnContainer* [▶ 1890]		Container with the descriptor matches (ContainerType_Vector_TcVnDMatch)
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (An appropriate color image will be created if required.)
aMatchColor	TcVnVector4_LREAL [▶ 1713]&	{-1, -1, -1, -1}	Color to draw the matches (for {-1, -1, -1, -1}, a random color is chosen for each point and line)
aSingleColor	TcVnVector4_LREAL [▶ 1713]&	{-1, -1, -1, -1}	Color to draw the single keypoints (for {-1, -1, -1, -1}, a random color is chosen for each point)
ipMatchesMask	ITcVnContainer* [▶ 1890]	nullptr	Mask to select the matches to be drawn (ContainerType_Vector_SINT; Set to 0 if all matches should be drawn.)
eFlags	ETcVnDrawMatchesFlags [▶ 1736]	DMF_DEFAULT	A combination of flags to support overdrawing an existing destination image and/or drawing additional (Rich-)Keypoint information (size and orientation) and/or skipping single keypoints

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.15 DrawOrientation

Draw the main axis of a set of points/contour based on a rotated rectangle.

Syntax

Definition:

```
HRESULT DrawOrientation(
    HRESULT               hrPrev,
    TcVnRotatedRectangle& stOrientation,
    ITcVnImage*           ipDestImage,
    TcVnVector4_LREAL&    aColorMainAxis,
    LONG                  nThickness
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stOrientation	TcVnRotatedRectangle [▶ 1782]&	Rotated rectangle, containig center of mass, lenghts of axes, and rotation angle in clockwise direction
ipDestImage	ITcVnImage* [▶ 1925]	Destination image
aColorMainAxis	TcVnVector4_LREAL [▶ 1713]&	Color of the major axis
nThickness	LONG	Line thickness

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.16 DrawOrientation (Exp)

Draw the main axis, the horizontal axis and the angle of a set of points/contour based on a rotated rectangle.

Syntax

Definition:

```
HRESULT DrawOrientation(
    HRESULT               hrPrev,
    TcVnRotatedRectangle& stOrientation,
    ITcVnImage*           ipDestImage,
```

```
    TcVnVector4_LREAL&    aColorMainAxis,
    LONG                  nThickness,
    TcVnVector4_LREAL&    aColorHorizontalAxis,
    double                fFontScale
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stOrientation	TcVnRotatedRectangle [▶ 1782]&	Rotated rectangle, containig center of mass, lenghts of axes, and rotation angle in clockwise direction
ipDestImage	ITcVnImage* [▶ 1925]	Destination image
aColorMainAxis	TcVnVector4_LREAL [▶ 1713]&	Color of the major axis
nThickness	LONG	Line thickness
aColorHorizontalAxis	TcVnVector4_LREAL [▶ 1713]&	Color of the horizontal axis
fFontScale	double	Scaling factor for the text

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.17 DrawPoint

Draw a point.

Syntax

Definition:

```
HRESULT DrawPoint(
    HRESULT             hrPrev,
    ULONG               nX,
    ULONG               nY,
    ITcVnImage*         ipDestImage,
    ETcVnDrawShape      eShape,
    TcVnVector4_LREAL&  aColor,
    ULONG               nSize = 3,
    LONG                nThickness = 1,
    ETcVnLineType       eLineType = LT_8_CONNECTED
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nX	ULONG		x coordinate of the point
nY	ULONG		y coordinate of the point
ipDestImage	ITcVnImage* [▶ 1925]		Destination image
eShape	ETcVnDrawShape [▶ 1737]		Point shape
aColor	TcVnVector4_LREAL [▶ 1713]&		Color (for {-1, -1, -1, -1}, a random color is chosen)
nSize	ULONG	3	Size of the shape (half width)
nThickness	LONG	1	Line thickness (if negative, the shape is filled if it is closed)
eLineType	ETcVnLineType [▶ 1745]	LT_8_CONNECTED	Line type

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.18 DrawPoints

Draw a collection of points.

Syntax

Definition:

```
HRESULT DrawPoints(
    HRESULT             hrPrev,
    ITcVnContainer*     ipPoints,
    ITcVnImage*         ipDestImage,
    ETcVnDrawShape      eShape,
    TcVnVector4_LREAL&  aColor,
    ULONG               nSize = 3,
    LONG                nThickness = 1,
    ETcVnLineType       eLineType = LT_8_CONNECTED
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipPoints	ITcVnContainer* [▶ 1890]		Container with TcVnPoint2_DINT or TcVnPoint2_REAL elements
ipDestImage	ITcVnImage* [▶ 1925]		Destination image
eShape	ETcVnDrawShape [▶ 1737]		Point shape
aColor	TcVnVector4_LREAL [▶ 1713]&		Color (for {-1, -1, -1, -1}, a random color is chosen for each point)
nSize	ULONG	3	Size of the shape (half width)
nThickness	LONG	1	Line thickness (if negative, the shape is filled if it is closed)
eLineType	ETcVnLineType [▶ 1745]	LT_8_CONNECTED	Line type

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.19 DrawPolygon

Draw a pologon using collection of points.

Syntax

Definition:

```
HRESULT DrawPolygon(
    HRESULT             hrPrev,
    ITcVnContainer*     ipPoints,
    ITcVnImage*         ipDestImage,
    TcVnVector4_LREAL&  aColor,
    LONG                nThickness,
    bool                bIsClosed = true,
    ETcVnLineType       eLineType = LT_8_CONNECTED,
    ULONG               nShift = 0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipPoints	ITcVnContainer* [▶ 1890]		Container with TcVnPoint2_DINT, TcVnPoint2_REAL, or TcVnPoint2_LREAL elements
ipDestImage	ITcVnImage* [▶ 1925]		Destination image
aColor	TcVnVector4_LREAL [▶ 1713]&		Color
nThickness	LONG		Line thickness (if negative, the shape is filled)
bIsClosed	bool	true	If it is true, the function draws a line from the last point to the first point to close the polygon.
eLineType	ETcVnLineType [▶ 1745]	LT_8_CONNECTED	Line type
nShift	ULONG	0	Fractional bits of the coordinates (bit shift)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.20 DrawRectangle

Draw a rectangle.

Syntax

Definition:

```
HRESULT DrawRectangle(
    HRESULT            hrPrev,
    ULONG              nTopLeftX,
    ULONG              nTopLeftY,
    ULONG              nBottomRightX,
    ULONG              nBottomRightY,
    ITcVnImage*        ipDestImage,
    TcVnVector4_LREAL& aColor,
    LONG               nThickness
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nTopLeftX	ULONG	x coordinate of the top left corner
nTopLeftY	ULONG	y coordinate of the top left corner
nBottomRightX	ULONG	x coordinate of the bottom right corner
nBottomRightY	ULONG	y coordinate of the bottom right corner
ipDestImage	ITcVnImage* [▶ 1925]	Destination image
aColor	TcVnVector4_LREAL [▶ 1713]&	Color
nThickness	LONG	Line thickness (if negative, the rectangle is filled)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.21 DrawRectangle (TcVnRectangle_DINT)

Draw a rectangle.

Syntax

Definition:

```
HRESULT DrawRectangle(
    HRESULT             hrPrev,
    TcVnRectangle_DINT& stRectangle,
    ITcVnImage*         ipDestImage,
    TcVnVector4_LREAL&  aColor,
    LONG                nThickness
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stRectangle	TcVnRectangle_DINT [▶ 1781]&	Rectangle to be drawn
ipDestImage	ITcVnImage* [▶ 1925]	Destination image
aColor	TcVnVector4_LREAL [▶ 1713]&	Color
nThickness	LONG	Line thickness (if negative, the rectangle is filled)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.22 DrawRectangle (TcVnRectangle_UDINT)

Draw a rectangle.

Syntax

Definition:

```
HRESULT DrawRectangle(
    HRESULT               hrPrev,
    TcVnRectangle_UDINT&  stRectangle,
    ITcVnImage*           ipDestImage,
    TcVnVector4_LREAL&    aColor,
    LONG                  nThickness
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stRectangle	TcVnRectangle_UDINT [▶ 1782]&	Rectangle to be drawn
ipDestImage	ITcVnImage* [▶ 1925]	Destination image
aColor	TcVnVector4_LREAL [▶ 1713]&	Color
nThickness	LONG	Line thickness (if negative, the rectangle is filled)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.23 DrawRotatedRectangle

Draw a rectangle.

Syntax

Definition:

```
HRESULT DrawRotatedRectangle(
    HRESULT                hrPrev,
    TcVnRotatedRectangle&  stRectangle,
    ITcVnImage*            ipDestImage,
    TcVnVector4_LREAL&     aColor,
    LONG                   nThickness,
    ETcVnLineType          eLineType = LT_8_CONNECTED
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stRectangle	TcVnRotatedRectangle [▶ 1782]&		Rectangle to be drawn (rotation angle in degrees)
ipDestImage	ITcVnImage* [▶ 1925]		Destination image
aColor	TcVnVector4_LREAL [▶ 1713]&		Color
nThickness	LONG		Line thickness (if negative, the rectangle is filled)
eLineType	ETcVnLineType [▶ 1745]	LT_8_CONNECTED	Line type

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.24 FillCircle

Paint a filled circle.

Syntax

Definition:

```
HRESULT FillCircle(
    HRESULT            hrPrev,
    ULONG              nCenterX,
    ULONG              nCenterY,
    ULONG              nRadius,
    ITcVnImage*        ipDestImage,
    TcVnVector4_LREAL& aColor
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nCenterX	ULONG	x coordinate of the center
nCenterY	ULONG	y coordinate of the center
nRadius	ULONG	Radius
ipDestImage	ITcVnImage* [▶ 1925]	Destination image
aColor	TcVnVector4_LREAL [▶ 1713]&	Color

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.25 FillContours

Paint a single point set or multiple point sets that are interpreted as contours.

Syntax

Definition:

```
HRESULT FillContours(
    HRESULT            hrPrev,
    ITcVnContainer*    ipContours,
    LONG               nContourIndex,
```

```
    ITcVnImage*         ipDestImage,
    TcVnVector4_LREAL&  aColor
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipContours	ITcVnContainer* [▶ 1890]	Single contour (ContainerType_Vector_TcVnPoint2_DINT) or multiple contours (ContainerType_Vector_Vector_TcVnPoint2_DINT)
nContourIndex	LONG	Index of a specific contour to be drawn (if negative, all contours within the container are drawn)
ipDestImage	ITcVnImage* [▶ 1925]	Destination image
aColor	TcVnVector4_LREAL [▶ 1713]&	Color

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.26 FillEllipse

Paint a filled ellipse.

Syntax

Definition:

```
HRESULT FillEllipse(
    HRESULT               hrPrev,
    TcVnRotatedRectangle& stEllipse,
    ITcVnImage*           ipDestImage,
    TcVnVector4_LREAL&    aColor
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stEllipse	TcVnRotatedRectangle [▶ 1782]&	Ellipse to be painted (rotation angle in degrees)
ipDestImage	ITcVnImage* [▶ 1925]	Destination image
aColor	TcVnVector4_LREAL [▶ 1713]&	Color

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.27 FillPolygon

Paint a filled pologon using a set of points.

Syntax

Definition:

```
HRESULT FillPolygon(
    HRESULT            hrPrev,
    ITcVnContainer*    ipPoints,
    ITcVnImage*        ipDestImage,
    TcVnVector4_LREAL& aColor
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipPoints	ITcVnContainer* [▶ 1890]	Container with TcVnPoint2_DINT, TcVnPoint2_REAL, or TcVnPoint2_LREAL elements
ipDestImage	ITcVnImage* [▶ 1925]	Destination image
aColor	TcVnVector4_LREAL [▶ 1713]&	Color

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.28 FillRectangle

Paint a filled rectangle.

Syntax

Definition:

```
HRESULT FillRectangle(
    HRESULT            hrPrev,
    ULONG              nTopLeftX,
    ULONG              nTopLeftY,
    ULONG              nBottomRightX,
    ULONG              nBottomRightY,
    ITcVnImage*        ipDestImage,
    TcVnVector4_LREAL& aColor
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nTopLeftX	ULONG	x coordinate of the top left corner
nTopLeftY	ULONG	y coordinate of the top left corner
nBottomRightX	ULONG	x coordinate of the bottom right corner
nBottomRightY	ULONG	y coordinate of the bottom right corner
ipDestImage	ITcVnImage* [▶ 1925]	Destination image
aColor	TcVnVector4_LREAL [▶ 1713]&	Color

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.29 FillRotatedRectangle

Paint a filled rectangle.

Syntax

Definition:

```
HRESULT FillRotatedRectangle(
    HRESULT               hrPrev,
    TcVnRotatedRectangle& stRectangle,
    ITcVnImage*           ipDestImage,
    TcVnVector4_LREAL&    aColor
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stRectangle	TcVnRotatedRectangle [▶ 1782]&	Rectangle to be painted (rotation angle in degrees)
ipDestImage	ITcVnImage* [▶ 1925]	Destination image
aColor	TcVnVector4_LREAL [▶ 1713]&	Color

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.30 PlotIntensityProfile

Plots the pixel intensity profile along a line segment in an image.

Syntax

Definition:

```
HRESULT PlotIntensityProfile(
    HRESULT            hrPrev,
    ITcVnImage*        ipSrcImage,
    ITcVnImage*&       ipDestImage,
    TcVnPoint2_REAL&   aStartPoint,
    TcVnPoint2_REAL&   aEndPoint,
    TcVnVector4_LREAL& aBackgroundColor = {0, 0, 0, 0},
    TcVnVector4_LREAL& aLineColor = {255, 255, 255, 255},
    TcVnVector4_LREAL& aGridColor = {64, 64, 64, 64},
    LONG               nScaleX = 2,
    LONG               nScaleY = 2,
    LONG               nThickness = 1,
    LONG               nDestChannels = 1
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (USINT, 1 channel)
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (USINT, 1 channel. x: position, y: intensity, origin: bottom left. An appropriate destination image will be created if required.)
aStartPoint	TcVnPoint2_REAL [▶ 1711]&		Start point of the line segment
aEndPoint	TcVnPoint2_REAL [▶ 1711]&		End point of the line segment
aBackgroundColor	TcVnVector4_LREAL [▶ 1713]&	{0, 0, 0, 0}	Background color
aLineColor	TcVnVector4_LREAL [▶ 1713]&	{255, 255, 255, 255}	Line color
aGridColor	TcVnVector4_LREAL [▶ 1713]&	{64, 64, 64, 64}	Grid color
nScaleX	LONG	2	Scale in x direction (ipDestImage will have a width of nScaleX * (ipSrcImage width - 1) + 1)
nScaleY	LONG	2	Scale in y direction (ipDestImage will have a height of nScaleY * 255 + 1)
nThickness	LONG	1	Line thickness
nDestChannels	LONG	1	ipDestImage channels (1 or 3)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.31 PutLabel

Write a label (text on unified background) into an image.

Syntax

Definition:

```
HRESULT PutLabel(
    HRESULT             hrPrev,
    PCCH                sText,
    ITcVnImage*         ipDestImage,
    ULONG               nX,
    ULONG               nY,
    double              fFontScale,
    ULONG               nThickness = 1,
    ETcVnFontType       eFontType = FT_HERSHEY_SIMPLEX,
    TcVnVector4_LREAL&  aFontColor = {0, 0, 0, 0},
    TcVnVector4_LREAL&  aBackgroundColor = {255, 255, 255, 255},
    ETcVnLineType       eLineType = LT_4_CONNECTED
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
sText	PCCH		Text
ipDestImage	ITcVnImage* [▶ 1925]		Destination image
nX	ULONG		x coordinate (bottom left)
nY	ULONG		y coordinate (bottom left)
fFontScale	double		Scaling factor
nThickness	ULONG	1	Line thickness
eFontType	ETcVnFontType [▶ 1742]	FT_HERSHEY_SIMPLEX	Font type
aFontColor	TcVnVector4_LREAL [▶ 1713]&	{0, 0, 0, 0}	Font color
aBackgroundColor	TcVnVector4_LREAL [▶ 1713]&	{255, 255, 255, 255}	Background color
eLineType	ETcVnLineType [▶ 1745]	LT_4_CONNECTED	Line type

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.11.32 PutText

Write text into an image.

Syntax

Definition:

```
HRESULT PutText(
    HRESULT             hrPrev,
    PCCH                sText,
    ITcVnImage*         ipDestImage,
    ULONG               nX,
    ULONG               nY,
    ETcVnFontType       eFontType,
    double              fFontScale,
    TcVnVector4_LREAL&  aColor,
    LONG                nThickness = 1,
    ETcVnLineType       eLineType = LT_8_CONNECTED,
    bool                bBottomLeftOrigin = false
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
sText	PCCH		Text
ipDestImage	ITcVnImage* [▶ 1925]		Destination image
nX	ULONG		x coordinate (bottom left)
nY	ULONG		y coordinate (bottom left)
eFontType	ETcVnFontType [▶ 1742]		Font type
fFontScale	double		Scaling factor
aColor	TcVnVector4_LREAL [▶ 1713]&		Text color
nThickness	LONG	1	Line thickness
eLineType	ETcVnLineType [▶ 1745]	LT_8_CONNECTED	Line type
bBottomLeftOrigin	bool	false	Sets the image origin to the bottom left corner, if true

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.12 Fourier Analysis

Diese Gruppe enthält Funktionen zur Frequenzanalyse und Frequenzfilterung in Bildern.

Funktionen

Butterworth-Filter

- CreateBandpassButterworthFilter [▶ 2412]
- CreateBandrejectButterworthFilter [▶ 2414]
- CreateHighpassButterworthFilter [▶ 2417]
- CreateLowpassButterworthFilter [▶ 2419]

Gauß-Filter

- CreateBandpassGaussianFilter [▶ 2413]
- CreateBandrejectGaussianFilter [▶ 2416]
- CreateHighpassGaussianFilter [▶ 2418]
- CreateLowpassGaussianFilter [▶ 2420]

Diskrete Fourier-Transformation

- Dft [▶ 2421]
- InverseDft [▶ 2422]
- OptimalDftSize [▶ 2423]

6.2.3.12.1 CreateBandpassButterworthFilter

Creates a bandpass Butterworth filter, which can be applied to a frequency domain image by element-wise multiplication.

Syntax

Definition:

```
HRESULT CreateBandpassButterworthFilter(
    HRESULT      hrPrev,
    ITcVnImage*& ipFilter,
    ULONG        nWidth,
    ULONG        nHeight,
    bool         bDoublePrecision,
    bool         bOriginAtCenter,
    double       fCutoffDistance,
    double       fBandWidth,
    ULONG        nOrder,
    double       fScale
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipFilter	ITcVnImage* [▶ 1925]&	Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).
nWidth	ULONG	Filter width (even, >= 2, must match the width of the image it is applied to)
nHeight	ULONG	Filter height (even, >= 2, must match the height of the image it is applied to)
bDoublePrecision	bool	If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).
bOriginAtCenter	bool	If true, the filter origin is shifted to the image center.
fCutoffDistance	double	The cutoff distance of the Butterworth filter (> 0).
fBandWidth	double	The band width of the Butterworth filter (> 0).
nOrder	ULONG	The order of the Butterworth filter (> 0).
fScale	double	The scale factor of the Butterworth filter denominator term (> 0, e.g. 0.414 or 1.0).

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.12.2 CreateBandpassGaussianFilter

Creates a bandpass Gaussian filter, which can be applied to a frequency domain image by element-wise multiplication.

Syntax

Definition:

```
HRESULT CreateBandpassGaussianFilter(
    HRESULT      hrPrev,
    ITcVnImage*& ipFilter,
    ULONG        nWidth,
```

```
    ULONG        nHeight,
    bool         bDoublePrecision,
    bool         bOriginAtCenter,
    double       fCutoffDistance,
    double       fBandWidth
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipFilter	ITcVnImage* [▶ 1925]&	Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).
nWidth	ULONG	Filter width (even, >= 2, must match the width of the image it is applied to)
nHeight	ULONG	Filter height (even, >= 2, must match the height of the image it is applied to)
bDoublePrecision	bool	If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).
bOriginAtCenter	bool	If true, the filter origin is shifted to the image center.
fCutoffDistance	double	The cutoff distance of the Gaussian filter (> 0).
fBandWidth	double	The band width of the Gaussian filter (> 0).

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.12.3 CreateBandrejectButterworthFilter

Creates a bandreject Butterworth filter, which can be applied to a frequency domain image by element-wise multiplication.

Syntax

Definition:

```
HRESULT CreateBandrejectButterworthFilter(
    HRESULT      hrPrev,
    ITcVnImage*& ipFilter,
```

```
    ULONG        nWidth,
    ULONG        nHeight,
    bool         bDoublePrecision,
    bool         bOriginAtCenter,
    double       fCutoffDistance,
    double       fBandWidth,
    ULONG        nOrder,
    double       fScale
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipFilter	ITcVnImage* [▶ 1925]&	Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).
nWidth	ULONG	Filter width (even, >= 2, must match the width of the image it is applied to)
nHeight	ULONG	Filter height (even, >= 2, must match the height of the image it is applied to)
bDoublePrecision	bool	If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).
bOriginAtCenter	bool	If true, the filter origin is shifted to the image center.
fCutoffDistance	double	The cutoff distance of the Butterworth filter (> 0).
fBandWidth	double	The band width of the Butterworth filter (> 0).
nOrder	ULONG	The order of the Butterworth filter (> 0).
fScale	double	The scale factor of the Butterworth filter denominator term (> 0, e.g. 0.414 or 1.0).

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.12.4 CreateBandrejectGaussianFilter

Creates a bandreject Gaussian filter, which can be applied to a frequency domain image by element-wise multiplication.

Syntax

Definition:

```
HRESULT CreateBandrejectGaussianFilter(
    HRESULT      hrPrev,
    ITcVnImage*& ipFilter,
    ULONG        nWidth,
    ULONG        nHeight,
    bool         bDoublePrecision,
    bool         bOriginAtCenter,
    double       fCutoffDistance,
    double       fBandWidth
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipFilter	ITcVnImage* [▶ 1925]&	Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).
nWidth	ULONG	Filter width (even, >= 2, must match the width of the image it is applied to)
nHeight	ULONG	Filter height (even, >= 2, must match the height of the image it is applied to)
bDoublePrecision	bool	If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).
bOriginAtCenter	bool	If true, the filter origin is shifted to the image center.
fCutoffDistance	double	The cutoff distance of the Gaussian filter (> 0).
fBandWidth	double	The band width of the Gaussian filter (> 0).

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.12.5 CreateHighpassButterworthFilter

Creates a highpass Butterworth filter, which can be applied to a frequency domain image by element-wise multiplication.

Syntax

Definition:

```
HRESULT CreateHighpassButterworthFilter(
    HRESULT      hrPrev,
    ITcVnImage*& ipFilter,
    ULONG        nWidth,
    ULONG        nHeight,
    bool         bDoublePrecision,
    bool         bOriginAtCenter,
    double       fCutoffDistance,
    ULONG        nOrder,
    double       fScale
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipFilter	ITcVnImage* [▶ 1925]&	Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).
nWidth	ULONG	Filter width (even, >= 2, must match the width of the image it is applied to)
nHeight	ULONG	Filter height (even, >= 2, must match the height of the image it is applied to)
bDoublePrecision	bool	If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).
bOriginAtCenter	bool	If true, the filter origin is shifted to the image center.
fCutoffDistance	double	The cutoff distance of the Butterworth filter (> 0).
nOrder	ULONG	The order of the Butterworth filter (> 0).
fScale	double	The scale factor of the Butterworth filter denominator term (> 0, e.g. 0.414 or 1.0).

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.12.6 CreateHighpassGaussianFilter

Creates a highpass Gaussian filter, which can be applied to a frequency domain image by element-wise multiplication.

Syntax

Definition:

```
HRESULT CreateHighpassGaussianFilter(
    HRESULT      hrPrev,
    ITcVnImage*& ipFilter,
    ULONG        nWidth,
    ULONG        nHeight,
    bool         bDoublePrecision,
    bool         bOriginAtCenter,
    double       fCutoffDistance
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipFilter	ITcVnImage* [▶ 1925]&	Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).
nWidth	ULONG	Filter width (even, >= 2, must match the width of the image it is applied to)
nHeight	ULONG	Filter height (even, >= 2, must match the height of the image it is applied to)
bDoublePrecision	bool	If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).
bOriginAtCenter	bool	If true, the filter origin is shifted to the image center.
fCutoffDistance	double	The cutoff distance of the Gaussian filter (> 0).

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.12.7 CreateLowpassButterworthFilter

Creates a lowpass Butterworth filter, which can be applied to a frequency domain image by element-wise multiplication.

Syntax

Definition:

```
HRESULT CreateLowpassButterworthFilter(
    HRESULT      hrPrev,
    ITcVnImage*& ipFilter,
    ULONG        nWidth,
    ULONG        nHeight,
    bool         bDoublePrecision,
    bool         bOriginAtCenter,
    double       fCutoffDistance,
    ULONG        nOrder,
    double       fScale
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipFilter	ITcVnImage* [▶ 1925]&	Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).
nWidth	ULONG	Filter width (even, >= 2, must match the width of the image it is applied to)
nHeight	ULONG	Filter height (even, >= 2, must match the height of the image it is applied to)
bDoublePrecision	bool	If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).
bOriginAtCenter	bool	If true, the filter origin is shifted to the image center.
fCutoffDistance	double	The cutoff distance of the Butterworth filter (> 0).
nOrder	ULONG	The order of the Butterworth filter (> 0).
fScale	double	The scale factor of the Butterworth filter denominator term (> 0, e.g. 0.414 or 1.0).

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.12.8 CreateLowpassGaussianFilter

Creates a lowpass Gaussian filter, which can be applied to a frequency domain image by element-wise multiplication.

Syntax

Definition:

```
HRESULT CreateLowpassGaussianFilter(
    HRESULT      hrPrev,
    ITcVnImage*& ipFilter,
    ULONG        nWidth,
    ULONG        nHeight,
    bool         bDoublePrecision,
    bool         bOriginAtCenter,
    double       fCutoffDistance
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipFilter	ITcVnImage* [▶ 1925]&	Returns the created filter (1 channel of type REAL or LREAL, depending on bDoublePrecision. Non-zero interface pointers are reused.).
nWidth	ULONG	Filter width (even, >= 2, must match the width of the image it is applied to)
nHeight	ULONG	Filter height (even, >= 2, must match the height of the image it is applied to)
bDoublePrecision	bool	If true, the filter is generated with double precision (LREAL) instead of single precision (REAL).
bOriginAtCenter	bool	If true, the filter origin is shifted to the image center.
fCutoffDistance	double	The cutoff distance of the Gaussian filter (> 0).

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.12.9 Dft

Computes the DFT for a given image.

Syntax

Definition:

```
HRESULT Dft(
    HRESULT      hrPrev,
    ITcVnImage*  ipSpatialImage,
    ITcVnImage*& ipFrequencyImage,
    bool         bPackedCCS,
    bool         bAutoPadding
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSpatialImage	ITcVnImage* [▶ 1925]	Source image (elements of type REAL or LREAL, 1 (Re) or 2 (Re + Im) channels)
ipFrequencyImage	ITcVnImage* [▶ 1925]&	Destination image (same type as ipSpatialImage, but number of channels and size can vary depending on bPackedCCS and bAutoPadding.
bPackedCCS	bool	If true and ipSpatialImage has only 1 channel, the result image will have 1 channel with packed complex-conjugate-symmetrical format results. Otherwise, the result image will have 2 separate channels (Re + Im), containing the full spectrum.
bAutoPadding	bool	If true, the input image is automatically padded (with 0s) to optimal size if required, to speed up dft (Creates a temporary copy so that ipSpatialImage content stays unchanged, which also requires some additional computation power. Therefore, it is recommended to compare execution times with and without padding.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.12.10 InverseDft

Compute the inverse DFT for a given frequency image.

Syntax

Definition:

```
HRESULT InverseDft(
    HRESULT      hrPrev,
    ITcVnImage*  ipFrequencyImage,
    ITcVnImage*& ipSpatialImage,
    bool         bRealOutput
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipFrequencyImage	ITcVnImage* [▶ 1925]	Source frequency image (elements of type REAL or LREAL, 1 (packed CCS) or 2 (Re + Im) channels)
ipSpatialImage	ITcVnImage* [▶ 1925]&	Destination image (Same type as ipFrequencyImage, but number of channels can vary depending on bRealOutput. An appropriate image will be created if required.)
bRealOutput	bool	Only relevant if ipFrequencyImage has 2 channels. If true, the result image will have only 1 channel. Otherwise, the result image will have 2 separate channels (Re + Im). If ipFrequencyImage has only 1 channel, packed CCS format is assumed and the result image will always have only 1 channel.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.12.11 OptimalDftSize

Computes the optimal image size for applying a dft (can lead to better performance).

Syntax

Definition:

```
HRESULT OptimalDftSize(
    HRESULT     hrPrev,
    ITcVnImage* ipImage,
    ULONG&      nOptWidth,
    ULONG&      nOptHeight
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]	Source image for which to compute the optimal width and height
nOptWidth	ULONG&	Returns the optimal width of the image
nOptHeight	ULONG&	Returns the optimal height of the image

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13 Geometric and Coordinate Transformations

Diese Gruppe enthält Funktionen zur Transformation von Bildern und Bildpunkten.

Funktionen

Calibration [▶ 2425]

- Intrinsische und extrinsische Kalibrierung
- Kompensation von optischen Verzeichnungen
- Transformation zwischen Bild- und Welt-Koordinaten

Affine Transformation

- ApplyRotationToAffineTransformation [▶ 2458]
- ApplyScalingToAffineTransformation [▶ 2459]
- ApplyTranslationToAffineTransformation [▶ 2460]
- ApplyYAxisInversionToAffineTransformation [▶ 2461]
- DecomposeAffineTransformation [▶ 2470]
- GenerateAffineTransformationUnitMatrix2D [▶ 2476]
- GetAffineTransformation [▶ 2476]
- GetAffineTransformation2D [▶ 2477]
- InvertAffineTransformation [▶ 2480]
- WarpAffine(Exp) [▶ 2488]
- WarpAffine_Container [▶ 2489]
- WarpAffine_Point [▶ 2490]
- WarpAffine_Rectangle [▶ 2491]

Transformation zwischen kartesischen und Polar-Koordinaten

- ConvertCartesianToPolarAngleImage [▶ 2462]
- ConvertCartesianToPolarAngles [▶ 2463]
- ConvertCartesianToPolarImages [▶ 2464]
- ConvertCartesianToPolarMagnitudeImage [▶ 2465]
- ConvertCartesianToPolarMagnitudes [▶ 2466]
- ConvertCartesianToPolarPoints [▶ 2467]
- ConvertPolarToCartesianImages [▶ 2468]
- ConvertPolarToCartesianPoints [▶ 2469]
- RemapImageToLogPolarSpace [▶ 2483]
- RemapImageToPolarSpace [▶ 2485]

Einfache Bildoperationen

- FlipImage [▶ 2475]
- PyramidDown [▶ 2482]
- PyramidUp [▶ 2483]
- ResizeImage [▶ 2486]

Bildausrichtung und Rotation

- AlignRotatedImageRegion [▶ 2457]
- RotateImage(Exp) [▶ 2487]

Perspektivische Transformation

- DecomposeHomography [▶ 2471]
- GetPerspectiveTransformation [▶ 2479]
- Homography [▶ 2479]
- PerspectiveTransformation [▶ 2481]
- WarpPerspective(Exp) [▶ 2493]
- WarpPerspective_Container [▶ 2493]

- WarpPerspective_Point [▶ 2494]
- WarpPerspective_Rectangle [▶ 2495]

6.2.3.13.1 Calibration

Diese Untergruppe enthält Funktionen zur Korrelation zwischen Bildpunkten und Echtweltpunkten. Sie werden üblicherweise zusammen mit den Vermessungsfunktionen [▶ 2650] verwendet.

Funktionen

Kalibration

- CalibrateCamera(Exp) [▶ 2426]
- CalibrateCameraManually [▶ 2431]
- CalibrateCameraPlanar [▶ 2433]
- CalibrateLinescanCamera [▶ 2435]
- DetectPatternPoints [▶ 2444]
- GenerateCalibrationPatternReferencePoints [▶ 2447]
- SortDetectedPatternPoints [▶ 2451]
- SolvePnP [▶ 2449]

Kompensation von Linsenverzeichnung

- CompensateLensDistortion [▶ 2438]
- CompensateLensDistortionForPoints [▶ 2441]

Transformation zwischen Bild- und Welt-Koordinaten

- TransformCoordinatesImageToWorld [▶ 2452]
- TransformCoordinatesWorldToImage [▶ 2456]
- TransformCoordinatesPlanar [▶ 2454]
- ImagePointsWorldDistance [▶ 2448]

Anforderungen an die Eingangsbilder

Die Eingangsbilder sollten nicht in einem gepackten oder gefilterten Format verwendet werden, wie z. B. im Bayer Format. Bei den Funktionen zur Kalibration hat das zur Folge, dass die Erkennung der Muster gestört wird, was zu Schlechteren oder gar keinen Kalibierergebnissen führen kann. Da bei den Kompensationsfunktionen eine Interpolation stattfindet, bei der die ursprüngliche Anordnung der Pixel verändert wird, kann das Bild im Anschluss nicht mehr richtig gewandelt werden. Daher wird empfohlen, eine Konvertierung immer zuerst durchzuführen, bevor eine Funktion aus dieser Untergruppe verwendet wird.

Anforderung an die Kameramatrix

$$CameraMatrix = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Die Kameramatrix setzt sich aus den Brennweiten fx und fy und dem Hauptpunkt (cx,cy) zusammen, die in Pixeleinheiten ausgedrückt werden. Der Hauptpunkt, ist der Schnittpunkt der optischen Achse mit dem Bild, liegt normalerweise in der Nähe der Bildmitte. Folgende Bedingungen müssen eingehalten werden:

- 0 < fx und fy
- cx >= 0 und <= Bildbreite -1
- cy >= 0 und <= Bildhöhe
- Die anderen Stellen der Matrix müssen, wie in der Abbildung dargestellt, mit 0 oder 1 gefüllt werden. Andere Werte sind nicht zulässig.

6.2.3.13.1.1 CalibrateCamera

Compute the camera parameters (intrinsic + extrinsic) by evaluating an image containing a calibration pattern (circles).
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT CalibrateCamera(
    HRESULT                       hrPrev,
    ITcVnImage*                   ipSrcImage,
    ITcVnContainer*               ipReferencePoints,
    TcVnMatrix3x3_LREAL&          aCameraMatrix,
    TcVnArray8_LREAL&             aDistortionCoefficients,
    TcVnMatrix3x3_LREAL&          aRotationMatrix,
    TcVnVector3_LREAL&            aTranslationVector,
    double&                       fReprojError,
    TcVnParamsBlobDetection&      stBlobDetectionParams,
    TcVnCameraCalibrationOptions& stCalibrationOptions
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image containing a calibration pattern (1 channel, USINT elements)
ipReferencePoints	ITcVnContainer* [▶ 1890]	Reference calibration pattern point positions (ContainerType_Vector_TcVnPoint3_REAL)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1713]&	Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 1713]&	Returns the translation vector
fReprojError	double&	Returns the reprojection error
stBlobDetectionParams	TcVnParamsBlobDetection [▶ 1771]&	Parameters for the internally used F_VN_DetectBlobs function
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 1764]&	Calibration options

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.1.2 CalibrateCamera (Exp)

Compute the camera parameters (intrinsic + extrinsic) by evaluating images containing a calibration pattern (circles). The extrinsic parameters are computed for the first image in the array.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT CalibrateCamera(
    HRESULT                     hrPrev,
    PVOID                       pSrcImages,
    USHORT                      nSrcArraySize,
    ITcVnContainer*             ipReferencePoints,
    TcVnMatrix3x3_LREAL&        aCameraMatrix,
    TcVnArray8_LREAL&           aDistortionCoefficients,
    TcVnMatrix3x3_LREAL&        aRotationMatrix,
    TcVnVector3_LREAL&          aTranslationVector,
    double&                     fReprojError,
```

```
    TcVnParamsBlobDetection&       stBlobDetectionParams,
    TcVnCameraCalibrationOptions& stCalibrationOptions,
    bool                           bSubpixelAccuracy = false,
    ETcVnEdgeDirection             eEdgeDirection = ED_DARK_TO_LIGHT,
    float                          fMinStrength = 50,
    ULONG                          nMaxThickness = 10,
    ULONG                          nSubpixelsIterations = 10,
    ULONG                          nSearchLines = 92,
    float                          fApproxPrecision = 0.001f,
    ETcVnEdgeDetectionAlgorithm    eAlgorithm = EDA_INTERPOLATION,
    ETcVnWorldCoordinateSystem     eWorldSystem = WCS_X_RIGHT_Y_DOWN
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
pSrcImages	PVOID		Pointer to an array of source images, each containing the same calibration pattern (1 channel, USINT elements)
nSrcArraySize	USHORT		pSrcImages array size
ipReferencePoints	ITcVnContainer* [▶ 1890]		Reference calibration pattern point positions (ContainerType_Vector_TcVnPoint3_REAL)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&		Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1713]&		Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&		Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 1713]&		Returns the translation vector
fReprojError	double&		Returns the reprojection error
stBlobDetectionParams	TcVnParamsBlobDetection [▶ 1771]&		Parameters for the internally used F_VN_DetectBlobs function
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 1764]&		Calibration options
bSubpixelAccuracy	bool	false	If true, the pattern points are detected with subpixel accuracy
eEdgeDirection	ETcVnEdgeDirection [▶ 1738]	ED_DARK_TO_LIGHT	Specification of the edge direction to search for (from center to outside ellipse)
fMinStrength	float	50	Specification of the minimum strength (intensity difference) of the edge to search for
nMaxThickness	ULONG	10	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels

Name	Type	Default	Description
nSubpixelsIterations	ULONG	10	Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)
nSearchLines	ULONG	92	Specifies the amount of search lines, which are equally distributed in all directions (must be >= 8 and a multiple of 4)
fApproxPrecision	float	0.001f	Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)
eAlgorithm	ETcVnEdgeDetectionAlgorithm [▶ 1738]	EDA_INTERPOLATION	Selection of the edge detection algorithm
eWorldSystem	ETcVnWorldCoordinateSystem [▶ 1759]	WCS_X_RIGHT_Y_DOWN	World coordinate system orientation

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.1.3 CalibrateCameraManually

Compute the camera parameters (intrinsic + extrinsic). The extrinsic parameters are computed for the first inner container.

Syntax

Definition:

```
HRESULT CalibrateCameraManually(
    HRESULT                     hrPrev,
    ITcVnContainer*             ipImagePoints,
    ULONG                       nImageWidth,
    ULONG                       nImageHeight,
    ITcVnContainer*             ipReferencePoints,
    TcVnMatrix3x3_LREAL&        aCameraMatrix,
    TcVnArray8_LREAL&           aDistortionCoefficients,
```

```
    TcVnMatrix3x3_LREAL&          aRotationMatrix,
    TcVnVector3_LREAL&            aTranslationVector,
    double&                       fReprojError,
    TcVnCameraCalibrationOptions& stCalibrationOptions,
    ETcVnWorldCoordinateSystem    eWorldSystem = WCS_X_RIGHT_Y_DOWN
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImagePoints	ITcVnContainer* [▶ 1890]		Provided image points of multiple images (ContainerType_Vector_Vector_TcVnPoint2_REAL, each inner container with at least 6 points). The container and point order must match ipReferencePoints.
nImageWidth	ULONG		Image width
nImageHeight	ULONG		Image height
ipReferencePoints	ITcVnContainer* [▶ 1890]		Reference world points (ContainerType_Vector_Vector_TcVnPoint3_REAL). The number of inner containers and their amount of points must match ipImagePoints.
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&		Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1713]&		Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&		Returns the rotation matrix (for the first inner container)
aTranslationVector	TcVnVector3_LREAL [▶ 1713]&		Returns the translation vector (for the first inner container)
fReprojError	double&		Returns the reprojection error
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 1764]&		Calibration options
eWorldSystem	ETcVnWorldCoordinateSystem [▶ 1759]	WCS_X_RIGHT_Y_DOWN	World coordinate system orientation

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.1.4 CalibrateCameraPlanar

Calibrate camera using a planar calibration pattern comprised of four circles marking the corners of a rectangle.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT CalibrateCameraPlanar(
    HRESULT                 hrPrev,
    ITcVnImage*             ipSrcImage,
    TcVnArray4_Point2_REAL& aMarkerPositions,
    TcVnMatrix3x3_LREAL&    aTransformationMatrix,
    TcVnArray4_Point2_REAL& aMarkerImagePositions,
    ULONG                   nMinMarkerArea = 100,
    ULONG                   nMaxMarkerArea = 10000,
    double                  fMaxMarkerEccentricity = 0.05
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Binary source image (background zero, markers non-zero)
aMarkerPositions	TcVnArray4_Point2_REAL [▶ 1713]&		Marker positions within their plane using world units (right-handed coordinate system)
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&		Returns a 3-by-3 matrix for the transformation from image coordinates into the reference coordinate system
aMarkerImagePositions	TcVnArray4_Point2_REAL [▶ 1713]&		Returns the marker positions in image coordinates (left-handed coordinate system)
nMinMarkerArea	ULONG	100	Minimum area for the detection of markers (Smaller markers are ignored.)
nMaxMarkerArea	ULONG	10000	Maximum area for the detection of markers (Larger markers are ignored.)
fMaxMarkerEccentricity	double	0.05	Maximum eccenticity for the detection of markers (The eccentricity measures deviations from a round shape (eccentricity = 0: round object; eccentricity = 1.0: linear object).)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.1.5 CalibrateLinescanCamera

Compute the camera parameters (intrinsic + extrinsic) by evaluating images containing a line calibration pattern. The pattern must consist of alternating vertical and diagonal lines, with each diagonal having the same constant slope: |\|\|...\|. The results are only valid for the x-direction of a line scan image. To get accurate x-values for a transformation into world coordinates, the y-coordinate must be set to half the sensor width and so equal to the cy-value of the camera matrix. For other y-values, the results are approximated and can be inaccurate. The extrinsic parameters are computed for the first image in the array.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT CalibrateLinescanCamera(
    HRESULT                       hrPrev,
    PVOID                         pSrcImages,
    USHORT                        nSrcArraySize,
    ITcVnContainer*               ipPatternLineOrigins,
    double                        fSlopeRad,
    TcVnMatrix3x3_LREAL&          aCameraMatrix,
    TcVnArray8_LREAL&             aDistortionCoefficients,
    TcVnMatrix3x3_LREAL&          aRotationMatrix,
    TcVnVector3_LREAL&            aTranslationVector,
    double&                       fReprojError,
    TcVnCameraCalibrationOptions& stCalibrationOptions,
    ETcVnEdgeDirection            eEdgeDirection,
    float                         fMinStrength = 100,
    ULONG                         nMaxSearchLines = 11,
    ULONG                         nMaxThickness = 5,
    ULONG                         nSubpixelsIterations = 50,
    float                         fApproxPrecision = 0.001f,
    ETcVnEdgeDetectionAlgorithm   eAlgorithm = EDA_APPROX_ERF
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
pSrcImages	PVOID		Pointer to an array of 1 or more source images (1 channel). If the images have more than 1 row, each row must contain an image of the same pattern position.
nSrcArraySize	USHORT		pSrcImages array size
ipPatternLineOrigins	ITcVnContainer* [▶ 1890]		X position of the pattern line origins (ContainerType_Vector_REAL, usually same origin for vertical and following diagonal line)
fSlopeRad	double		Slope of the diagonal line in rad (> 0, < PI/2), relative to the vertical line (i.e., 0 would be a vertical line, PI/2 a horizontal line)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&		Returns the camera matrix
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1713]&		Returns the lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&		Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 1713]&		Returns the translation vector
fReprojError	double&		Returns the reprojection error
stCalibrationOptions	TcVnCameraCalibrationOptions [▶ 1764]&		Calibration options (recommended to set bFixAspectRatio, bFixPrincipalPoint, bZeroTangentDist)
eEdgeDirection	ETcVnEdgeDirection [▶ 1738]		Specification of the edge direction to search for
fMinStrength	float	100	Specification of the minimum strength (intensity difference) of the edge to search for
nMaxSearchLines	ULONG	11	Maximum number of search lines (equally distributed over the image height, at most 1 searchline per image row)

Name	Type	Default	Description
nMaxThickness	ULONG	5	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels
nSubpixelsIterations	ULONG	50	Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)
fApproxPrecision	float	0.001f	Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)
eAlgorithm	ETcVnEdgeDetectionAlgorithm [▶ 1738]	EDA_APPROX_ERF	Selection of the edge detection algorithm

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.1.6 CompensateLensDistortion

Transforms an image to compensate the lens distortion.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT CompensateLensDistortion(
    HRESULT              hrPrev,
    ITcVnImage*          ipSrcImage,
    ITcVnImage*&         ipDestImage,
    TcVnMatrix3x3_LREAL& aCameraMatrix,
    TcVnArray4_LREAL&    aDistortionCoefficients
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray4_LREAL [▶ 1713]&	Lens distortion coefficients [k1, k2, p1, p2]

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.1.7 CompensateLensDistortion (Exp1)

Transforms an image to compensate the lens distortion.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT CompensateLensDistortion(
    HRESULT              hrPrev,
    ITcVnImage*          ipSrcImage,
    ITcVnImage*&         ipDestImage,
    TcVnMatrix3x3_LREAL& aCameraMatrix,
    TcVnArray8_LREAL&    aDistortionCoefficients
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1713]&	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.1.8 CompensateLensDistortion (Exp2)

Transforms an image to compensate the lens distortion.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT CompensateLensDistortion(
    HRESULT              hrPrev,
    ITcVnImage*          ipSrcImage,
    ITcVnImage*&         ipDestImage,
    TcVnMatrix3x3_LREAL& aCameraMatrix,
    TcVnArray8_LREAL&    aDistortionCoefficients,
    TcVnMatrix3x3_LREAL& aNewCameraMatrix
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1713]&	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aNewCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	Allows additional scaling and shifting of the result image

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.1.9 CompensateLensDistortionForPoints

Transforms point coordinates to compensate the lens distortion.

Syntax

Definition:

```
HRESULT CompensateLensDistortionForPoints(
    HRESULT              hrPrev,
    ITcVnContainer*      ipSrcPoints,
    ITcVnContainer*&     ipDestPoints,
    TcVnMatrix3x3_LREAL& aCameraMatrix,
    TcVnArray4_LREAL&    aDistortionCoefficients
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPoints	ITcVnContainer* [▶ 1890]	Container with source point coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_DINT)
ipDestPoints	ITcVnContainer* [▶ 1890]&	Returns a container with the transformed point coordinates (ContainerType_Vector_TcVnPoint2_REAL)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray4_LREAL [▶ 1713]&	Lens distortion coefficients [k1, k2, p1, p2]

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.1.10 CompensateLensDistortionForPoints (Exp1)

Transforms point coordinates to compensate the lens distortion.

Syntax

Definition:

```
HRESULT CompensateLensDistortionForPoints(
    HRESULT              hrPrev,
    ITcVnContainer*      ipSrcPoints,
    ITcVnContainer*&     ipDestPoints,
    TcVnMatrix3x3_LREAL& aCameraMatrix,
    TcVnArray8_LREAL&    aDistortionCoefficients
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPoints	ITcVnContainer* [▶ 1890]	Container with source point coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_DINT)
ipDestPoints	ITcVnContainer* [▶ 1890]&	Returns a container with the transformed point coordinates (ContainerType_Vector_TcVnPoint2_REAL)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1713]&	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.1.11 CompensateLensDistortionForPoints (Exp2)

Transforms point coordinates to compensate the lens distortion.

Syntax

Definition:

```
HRESULT CompensateLensDistortionForPoints(
    HRESULT              hrPrev,
    ITcVnContainer*      ipSrcPoints,
    ITcVnContainer*&     ipDestPoints,
    TcVnMatrix3x3_LREAL& aCameraMatrix,
    TcVnArray8_LREAL&    aDistortionCoefficients,
    TcVnMatrix3x3_LREAL& aNewCameraMatrix
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPoints	ITcVnContainer* [▶ 1890]	Container with source point coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_DINT)
ipDestPoints	ITcVnContainer* [▶ 1890]&	Returns a container with the transformed point coordinates (ContainerType_Vector_TcVnPoint2_REAL)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	Camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1713]&	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aNewCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	Allows additional shifting of the result point coordinates

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.1.12 DetectPatternPoints

Detects calibration pattern points (circles) within the provided image.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT DetectPatternPoints(
    HRESULT                       hrPrev,
    ITcVnImage*                   ipSrcImage,
    ITcVnContainer*&              ipPatternPoints,
    TcVnParamsBlobDetection&      stBlobDetectionParams,
    ULONG                         nNumberOfPoints,
    float                         fSizeDiffThresFactor = 0.3f,
    bool                          bSubpixelAccuracy = false,
    ETcVnEdgeDirection            eEdgeDirection = ED_DARK_TO_LIGHT,
    float                         fMinStrength = 50,
    ULONG                         nMaxThickness = 10,
    ULONG                         nSubpixelsIterations = 10,
    ULONG                         nSearchLines = 92,
    float                         fApproxPrecision = 0.001f,
    ETcVnEdgeDetectionAlgorithm   eAlgorithm = EDA_INTERPOLATION
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (USINT, 1 channel)
ipPatternPoints	ITcVnContainer* [▶ 1890]&		Returns the pattern point positions (ContainerType_Vector_TcVnPoint2_REAL; Non-zero interface pointers are reused.)
stBlobDetectionParams	TcVnParamsBlobDetection [▶ 1771]&		Parameters to detect and filter contours.
nNumberOfPoints	ULONG		Expected number of pattern points
fSizeDiffThresFactor	float	0.3f	Threshold for filtering contours by average size
bSubpixelAccuracy	bool	false	If true, the pattern points are detected with subpixel accuracy
eEdgeDirection	ETcVnEdgeDirection [▶ 1738]	ED_DARK_TO_LIGHT	Specification of the edge direction to search for (from center to outside circle)
fMinStrength	float	50	Specification of the minimum strength (intensity difference) of the edge to search for
nMaxThickness	ULONG	10	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels
nSubpixelsIterations	ULONG	10	Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)
nSearchLines	ULONG	92	Specifies the amount of search lines, which are equally distributed in all directions (must be >= 8 and a multiple of 4)

Name	Type	Default	Description
fApproxPrecision	float	0.001f	Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)
eAlgorithm	ETcVnEdgeDetectionAlgorithm [▶ 1738]	EDA_INTERPOLATION	Selection of the edge detection algorithm

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.1.13 GenerateCalibrationPatternReferencePoints

Generate reference points for common calibration patterns.

Syntax

Definition:

```
HRESULT GenerateCalibrationPatternReferencePoints(
    HRESULT                       hrPrev,
    ITcVnContainer*&              ipReferencePoints,
    ETcVnCalibrationPattern       ePattern,
    ETcVnCalibrationPatternOrigin eOrigin,
    ULONG                         nWidth,
    ULONG                         nHeight,
    float                         fDistX,
    float                         fDistY
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipReferencePoints	ITcVnContainer* [▶ 1890]&	Returns the generated reference points (ContainerType_Vector_TcVnPoint3_REAL)
ePattern	ETcVnCalibrationPattern [▶ 1717]	Calibration pattern type
eOrigin	ETcVnCalibrationPatternOrigin [▶ 1717]	Calibration pattern origin
nWidth	ULONG	Pattern width, i.e. number of pattern points in x direction
nHeight	ULONG	Pattern height, i.e. number of pattern points in y direction
fDistX	float	Distance between pattern points in x direction
fDistY	float	Distance between pattern points in y direction

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.1.14 ImagePointsWorldDistance

Computes the distance in the world coordinate system between two points (in the same world coordinate z layer) given in image coordinates.

Syntax

Definition:

```
HRESULT ImagePointsWorldDistance(
    HRESULT              hrPrev,
    TcVnPoint2_REAL&     aImagePoint1,
    TcVnPoint2_REAL&     aImagePoint2,
    double&              fWorldDistance,
    TcVnMatrix3x3_LREAL& aCameraMatrix,
    TcVnArray8_LREAL&    aDistortionCoefficients,
    TcVnMatrix3x3_LREAL& aRotationMatrix,
    TcVnVector3_LREAL&   aTranslationVector,
    double               fZ
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aImagePoint1	TcVnPoint2_REAL [▶ 1711]&	First point in the image coordinate system
aImagePoint2	TcVnPoint2_REAL [▶ 1711]&	Second point in the image coordinate system
fWorldDistance	double&	Returns the distance between aImagePoint1 and aImagePoint2 in the world coordinate system
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	3x3 camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1713]&	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 1713]&	Translation vector
fZ	double	z coordinate (world coordinate system) of the given points (0 would be at (toplevel) calibration pattern)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.1.15 SolvePnP

Computes the rotation matrix and translation vector from 3D-2D point correspondences.

Syntax

Definition:

```
HRESULT SolvePnP(
    HRESULT              hrPrev,
    ITcVnContainer*      ipImagePoints,
    ITcVnContainer*      ipReferencePoints,
    TcVnMatrix3x3_LREAL& aCameraMatrix,
    TcVnArray8_LREAL&    aDistortionCoefficients,
    TcVnMatrix3x3_LREAL& aRotationMatrix,
    TcVnVector3_LREAL&   aTranslationVector,
    double&              fReprojError,
    ETcVnSolvePnPMethod  eMethod = SPM_ITERATIVE,
    bool                 bUseExtrinsicGuess = false
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImagePoints	ITcVnContainer* [▶ 1890]		Image points (ContainerType_Vector_TcVnPoint2_REAL, requires at least 4 points in the same order as in ipReferencePoints)
ipReferencePoints	ITcVnContainer* [▶ 1890]		Reference world points (ContainerType_Vector_TcVnPoint3_REAL, same number of points as in ipImagePoints)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&		Camera matrix (e.g. computed with one of the CalibrateCamera functions)
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1713]&		Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6] (e.g. computed with one of the CalibrateCamera functions)
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&		Returns the rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 1713]&		Returns the translation vector
fReprojError	double&		Returns the reprojection error
eMethod	ETcVnSolvePnPMethod [▶ 1753]	SPM_ITERATIVE	Solve PnP method
bUseExtrinsicGuess	bool	false	If true, the provided rotation matrix and translation vector are used as an initial guess for further optimization (only supported for the SPM_ITERATIVE method).

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.1.16 SortDetectedPatternPoints

Sort the detected pattern points according to the relative positions of the calibration pattern points.

Syntax

Definition:

```
HRESULT SortDetectedPatternPoints(
    HRESULT         hrPrev,
    ITcVnContainer* ipImagePoints,
    ITcVnContainer* ipPatternPoints
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImagePoints	ITcVnContainer* [▶ 1890]	Detected pattern point positions in the image coordinate system (will be sorted by this function, ContainerType_Vector_TcVnPoint2_REAL)
ipPatternPoints	ITcVnContainer* [▶ 1890]	The pattern points in the calibration pattern coordinate system, used as a reference to sort ipImagePoints (ContainerType_Vector_TcVnPoint3_REAL)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.1.17 TransformCoordinatesImageToWorld (Container)

Transform 2D image point coordinates to 3D world coordinates (using intrinsic + extrinsic calibration results).

Syntax

Definition:

```
HRESULT TransformCoordinatesImageToWorld(
    HRESULT              hrPrev,
    ITcVnContainer*      ipSrcPoints2D,
    ITcVnContainer*&     ipDestPoints3D,
    TcVnMatrix3x3_LREAL& aCameraMatrix,
    TcVnArray8_LREAL&    aDistortionCoefficients,
    TcVnMatrix3x3_LREAL& aRotationMatrix,
    TcVnVector3_LREAL&   aTranslationVector,
    double               fZ
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPoints2D	ITcVnContainer* [▶ 1890]	Container with 2D source points (TcVnPoint2_DINT or TcVnPoint2_REAL or TcVnPoint2_LREAL)
ipDestPoints3D	ITcVnContainer* [▶ 1890]&	Returns the transformed 3D points (TcVnPoint3_LREAL)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	3x3 camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1713]&	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 1713]&	Translation vector
fZ	double	z coordinate (world coordinate system) of the transformed points (0 would be at (toplevel) calibration pattern)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.1.18 TransformCoordinatesImageToWorld (Point)

Transform 2D image point coordinate to 3D world coordinate (using intrinsic + extrinsic calibration results).

Syntax

Definition:

```
HRESULT TransformCoordinatesImageToWorld(
    HRESULT              hrPrev,
    TcVnPoint2_LREAL&    aSrcPoint,
    TcVnPoint3_LREAL&    aDestPoint,
    TcVnMatrix3x3_LREAL& aCameraMatrix,
```

```
    TcVnArray8_LREAL&     aDistortionCoefficients,
    TcVnMatrix3x3_LREAL&  aRotationMatrix,
    TcVnVector3_LREAL&    aTranslationVector,
    double                fZ
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aSrcPoint	TcVnPoint2_LREAL [▶ 1711]&	Point in the image coordinate system
aDestPoint	TcVnPoint3_LREAL [▶ 1711]&	Returns the point in the world coordinate system
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	3x3 camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1713]&	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 1713]&	Translation vector
fZ	double	z coordinate (world coordinate system) of the transformed points (0 would be at (toplevel) calibration pattern)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.1.19 TransformCoordinatesPlanar (Container)

Compute real-world coordinates for given image points or vice versa.

Syntax

Definition:

```
HRESULT TransformCoordinatesPlanar(
    HRESULT              hrPrev,
    ITcVnContainer*      ipSrcPoints,
    ITcVnContainer*&     ipDestPoints,
    TcVnMatrix3x3_LREAL& aTransformationMatrix
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPoints	ITcVnContainer* [▶ 1890]	Container with 2D source points (TcVnPoint2_DINT or TcVnPoint2_REAL or TcVnPoint2_LREAL)
ipDestPoints	ITcVnContainer* [▶ 1890]&	Returns the transformed 2D points (TcVnPoint2_LREAL)
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	Transformation matrix obtained with VnCalibrateCameraPlanar() or its inverse matrix obtained with VnInvertMatrix3x3()

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.1.20 TransformCoordinatesPlanar (Point)

Compute real-world coordinates for a given image point or vice versa.

Syntax

Definition:

```
HRESULT TransformCoordinatesPlanar(
    HRESULT              hrPrev,
    TcVnPoint2_LREAL&    aSrcPoint,
    TcVnPoint2_LREAL&    aDestPoint,
    TcVnMatrix3x3_LREAL& aTransformationMatrix
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aSrcPoint	TcVnPoint2_LREAL [▶ 1711]&	Point in the source coordinate system
aDestPoint	TcVnPoint2_LREAL [▶ 1711]&	Returns point in the destination coordinate system
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	Transformation matrix obtained with VnCalibrateCameraPlanar() or its inverse matrix obtained with VnInvertMatrix3x3()

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.1.21 TransformCoordinatesWorldToImage (Container)

Transform 3D world coordinates to 2D image coordinates (using intrinsic + extrinsic calibration results).

Syntax

Definition:

```
HRESULT TransformCoordinatesWorldToImage(
    HRESULT              hrPrev,
    ITcVnContainer*      ipSrcPoints3D,
    ITcVnContainer*&     ipDestPoints2D,
    TcVnMatrix3x3_LREAL& aCameraMatrix,
    TcVnArray8_LREAL&    aDistortionCoefficients,
    TcVnMatrix3x3_LREAL& aRotationMatrix,
    TcVnVector3_LREAL&   aTranslationVector,
    bool                 bInteger
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPoints3D	ITcVnContainer* [▶ 1890]	Container with 3D source points (TcVnPoint3_LREAL)
ipDestPoints2D	ITcVnContainer* [▶ 1890]&	Returns the transformed 2D points (TcVnPoint2_REAL or TcVnPoint2_DINT, depending on bInteger)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	3x3 camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1713]&	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 1713]&	Translation vector
bInteger	bool	Selects whether the function returns a container of TcVnPoint2_DINT or TcVnPoint2_REAL

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.1.22 TransformCoordinatesWorldToImage (Point)

Transform 3D world coordinates to 2D image coordinates (using intrinsic + extrinsic calibration results).

Syntax

Definition:

```
HRESULT TransformCoordinatesWorldToImage(
    HRESULT              hrPrev,
    TcVnPoint3_LREAL&    aSrcPoint,
    TcVnPoint2_REAL&     aDestPoint,
    TcVnMatrix3x3_LREAL& aCameraMatrix,
    TcVnArray8_LREAL&    aDistortionCoefficients,
    TcVnMatrix3x3_LREAL& aRotationMatrix,
    TcVnVector3_LREAL&   aTranslationVector
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aSrcPoint	TcVnPoint3_LREAL [▶ 1711]&	3D source point (world coordinate system)
aDestPoint	TcVnPoint2_REAL [▶ 1711]&	Returns the transformed 2D point (image coordinate system)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	3x3 camera matrix containing the intrinsic parameters
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1713]&	Lens distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 1713]&	Translation vector

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.2 AlignRotatedImageRegion

Copies the provided image region and aligns it to the image axes.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT AlignRotatedImageRegion(
    HRESULT                         hrPrev,
    ITcVnImage*                     ipSrcImage,
    ITcVnImage*&                    ipDestImage,
    TcVnRotatedRectangle&           stRegion,
    ETcVnInterpolationType          eInterpolationType = IT_BILINEAR,
    ETcVnBorderInterpolationMethod eBorderInterpolation = BIM_CONSTANT,
    TcVnVector4_LREAL&              aBorderValue = {0, 0, 0, 0}
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (An appropriate image will be created if required.)
stRegion	TcVnRotatedRectangle [▶ 1782]&		Image region to be aligned
eInterpolationType	ETcVnInterpolationType [▶ 1744]	IT_BILINEAR	Interpolation method
eBorderInterpolation	ETcVnBorderInterpolationMethod [▶ 1716]	BIM_CONSTANT	Border interpolation method (ISOLATED not supported)
aBorderValue	TcVnVector4_LREAL [▶ 1713]&	{0, 0, 0, 0}	Border value, if CONSTANT is used

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.3 ApplyRotationToAffineTransformation

Apply a rotation to an existing 2D affine transformation matrix.

Syntax

Definition:

```
HRESULT ApplyRotationToAffineTransformation(
    HRESULT              hrPrev,
    TcVnMatrix2x3_LREAL& aTransformationMatrix,
    double               fAngle,
    bool                 bUsePreMultiplication = true
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1713]&		Affine transformation matrix to which the rotation is applied
fAngle	double		Angle in radians (positive means counter-clockwise)
bUsePreMultiplication	bool	true	Select if pre- or post-multiplication should be used

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.4 ApplyScalingToAffineTransformation

Apply a scaling to an existing 2D affine transformation matrix.

Syntax

Definition:

```
HRESULT ApplyScalingToAffineTransformation(
    HRESULT              hrPrev,
    TcVnMatrix2x3_LREAL& aTransformationMatrix,
    double               fScaleX,
    double               fScaleY,
    bool                 bUsePreMultiplication = true
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1713]&		Affine transformation matrix to which the scaling is applied
fScaleX	double		Scaling factor in x direction
fScaleY	double		Scaling factor in y direction
bUsePreMultiplication	bool	true	Select if pre- or post-multiplication should be used

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.5 ApplyTranslationToAffineTransformation

Apply a translation to an existing 2D affine transformation matrix.

Syntax

Definition:

```
HRESULT ApplyTranslationToAffineTransformation(
    HRESULT              hrPrev,
    TcVnMatrix2x3_LREAL& aTransformationMatrix,
    double               fDeltaX,
    double               fDeltaY,
    bool                 bUsePreMultiplication = true
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1713]&		Affine transformation matrix to which the translation is applied
fDeltaX	double		Translation in x direction
fDeltaY	double		Translation in y direction
bUsePreMultiplication	bool	true	Select if pre- or post-multiplication should be used

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.6 ApplyYAxisInversionToAffineTransformation

Apply an inversion of the y-axis direction to an existing 2D affine transformation matrix.

Syntax

Definition:

```
HRESULT ApplyYAxisInversionToAffineTransformation(
    HRESULT              hrPrev,
    TcVnMatrix2x3_LREAL& aTransformationMatrix,
    bool                 bUsePreMultiplication = true
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1713]&		Affine transformation matrix to which the inversion of the y-axis direction is applied
bUsePreMultiplication	bool	true	Select if pre- or post-multiplication should be used

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.7 ConvertCartesianToPolarAngleImage

Converts cartesian coordinates (x, y) to polar angle.

Syntax

Definition:

```
HRESULT ConvertCartesianToPolarAngleImage(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImageX,
    ITcVnImage*  ipSrcImageY,
    ITcVnImage*& ipDestImageAng,
    bool         bAngleInDegrees = false
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImageX	ITcVnImage* [▶ 1925]		Source image containing the x values (elements of type REAL or LREAL)
ipSrcImageY	ITcVnImage* [▶ 1925]		Source image containing the y values (Same element type as ipSrcImageX)
ipDestImageAng	ITcVnImage* [▶ 1925]&		Destination image containing the angles (Same element type as source images. An appropriate destination image will be created if required.)
bAngleInDegrees	bool	false	Specifys, if the angles should be in degrees or radians

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.8 ConvertCartesianToPolarAngles

Converts cartesian coordinates (x, y) to polar angle.

Syntax

Definition:

```
HRESULT ConvertCartesianToPolarAngles(
    HRESULT          hrPrev,
    ITcVnContainer*  ipSrcPointsCart,
    ITcVnContainer*& ipDestAngles,
    bool             bAngleInDegrees = false
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPointsCart	ITcVnContainer* [▶ 1890]		Container with cartesian points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestAngles	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the angles (ContainerType_Vector_REAL, if source points are of type TcVnPoint2_DINT or TcVnPoint2_REAL; else ContainerType_Vector_LREAL. Non-zero interface pointers are reused.)
bAngleInDegrees	bool	false	Specify, if the angles should be in degrees or radians

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.9 ConvertCartesianToPolarImages

Converts cartesian coordinates (x, y) to polar coordinates (magnitude, angle).

Syntax

Definition:

```
HRESULT ConvertCartesianToPolarImages(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImageX,
    ITcVnImage*  ipSrcImageY,
    ITcVnImage*& ipDestImageMag,
    ITcVnImage*& ipDestImageAng,
    bool         bAngleInDegrees = false
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImageX	ITcVnImage* [▶ 1925]		Source image containing the x values (elements of type REAL or LREAL)
ipSrcImageY	ITcVnImage* [▶ 1925]		Source image containing the y values (Same element type as ipSrcImageX)
ipDestImageMag	ITcVnImage* [▶ 1925]&		Destination image containing the magnitudes (Same element type as source images. An appropriate destination image will be created if required.)
ipDestImageAng	ITcVnImage* [▶ 1925]&		Destination image containing the angles (Same element type as source images. An appropriate destination image will be created if required.)
bAngleInDegrees	bool	false	Specifys, if the angles should be in degrees or radians

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.10 ConvertCartesianToPolarMagnitudeImage

Converts cartesian coordinates (x, y) to polar magnitude.

Syntax

Definition:

```
HRESULT ConvertCartesianToPolarMagnitudeImage(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImageX,
```

```
    ITcVnImage*  ipSrcImageY,
    ITcVnImage*& ipDestImageMag
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImageX	ITcVnImage* [▶ 1925]	Source image containing the x values (elements of type REAL or LREAL)
ipSrcImageY	ITcVnImage* [▶ 1925]	Source image containing the y values (Same element type as ipSrcImageX)
ipDestImageMag	ITcVnImage* [▶ 1925]&	Destination image containing the magnitudes (Same element type as source images. An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.11 ConvertCartesianToPolarMagnitudes

Converts cartesian coordinates (x, y) to polar magnitude.

Syntax

Definition:

```
HRESULT ConvertCartesianToPolarMagnitudes(
    HRESULT          hrPrev,
    ITcVnContainer*  ipSrcPointsCart,
    ITcVnContainer*& ipDestMags
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPointsCart	ITcVnContainer* [▶ 1890]	Container with cartesian points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestMags	ITcVnContainer* [▶ 1890]&	Returns a container which is filled with the magnitudes (ContainerType_Vector_REAL, if source points are of type TcVnPoint2_DINT or TcVnPoint2_REAL; else ContainerType_Vector_LREAL. Non-zero interface pointers are reused.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.12 ConvertCartesianToPolarPoints

Converts cartesian coordinates (x, y) to polar coordinates (magnitude, angle).

Syntax

Definition:

```
HRESULT ConvertCartesianToPolarPoints(
    HRESULT          hrPrev,
    ITcVnContainer*  ipSrcPointsCart,
    ITcVnContainer*& ipDestPointsPolar,
    bool             bAngleInDegrees = false
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPointsCart	ITcVnContainer* [▶ 1890]		Container with cartesian points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestPointsPolar	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the converted points in polar coordinates (ContainerType_Vector_TcVnPoint2_REAL, if source points are of type TcVnPoint2_DINT or TcVnPoint2_REAL; else ContainerType_Vector_TcVnPoint2_LREAL. Non-zero interface pointers are reused.)
bAngleInDegrees	bool	false	Specifys, if the angles should be in degrees or radians

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.13 ConvertPolarToCartesianImages

Converts polar coordinates (magnitude, angle) to cartesian coordinates (x, y).

Syntax

Definition:

```
HRESULT ConvertPolarToCartesianImages(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImageMag,
    ITcVnImage*  ipSrcImageAng,
    ITcVnImage*& ipDestImageX,
```

```
    ITcVnImage*& ipDestImageY,
    bool         bAngleInDegrees = false
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImageMag	ITcVnImage* [▶ 1925]		Source image containing the magnitudes (elements of type REAL or LREAL)
ipSrcImageAng	ITcVnImage* [▶ 1925]		Source image containing the angles (Same element type as ipSrcImageMag)
ipDestImageX	ITcVnImage* [▶ 1925]&		Destination image containing the x values (Same element type as source images. An appropriate destination image will be created if required.)
ipDestImageY	ITcVnImage* [▶ 1925]&		Destination image containing the y values (Same element type as source images. An appropriate destination image will be created if required.)
bAngleInDegrees	bool	false	Specifys, if the angles are in degrees or radians

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.14 ConvertPolarToCartesianPoints

Converts polar coordinates (magnitude, angle) to cartesian coordinates (x, y).

Syntax

Definition:

```
HRESULT ConvertPolarToCartesianPoints(
    HRESULT          hrPrev,
    ITcVnContainer*  ipSrcPointsPolar,
```

```
    ITcVnContainer*& ipDestPointsCart,
    bool             bAngleInDegrees = false
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPointsPolar	ITcVnContainer* [▶ 1890]		Container with polar coordinates (ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestPointsCart	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the converted points in cartesian coordinates (same type ID as ipSrcPointsPolar. Non-zero interface pointers are reused.)
bAngleInDegrees	bool	false	Specifys, if the angles are in degrees or radians

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.15 DecomposeAffineTransformation

Decompose an affine transformation matrix and return the rotation angle (clockwise) in radian, translation vector (pixels), scale, and skew vectors (angle in radian). The decomposition method assumes the following order of transformation: Translation, Rotation, Scale, and Skew.

Syntax

Definition:

```
HRESULT DecomposeAffineTransformation(
    HRESULT              hrPrev,
    TcVnMatrix2x3_LREAL& aAffineTransform,
    double&              fRotationAngle,
    TcVnVector2_LREAL&   aTranslationVector = {0, 0},
    TcVnVector2_LREAL&   aScaleVector = {0, 0},
    TcVnVector2_LREAL&   aSkewVector = {0, 0}
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aAffineTransform	TcVnMatrix2x3_LREAL [▶ 1713]&		Affine transformation matrix
fRotationAngle	double&		Rotation angle (radian) in the clockwise direction. The rotation center is the origin point (0,0)
aTranslationVector	TcVnVector2_LREAL [▶ 1713]&	{0, 0}	2D translation vector. The values are in pixels
aScaleVector	TcVnVector2_LREAL [▶ 1713]&	{0, 0}	2D scale vector
aSkewVector	TcVnVector2_LREAL [▶ 1713]&	{0, 0}	2D skew vector. The values are the skew angles in x and y directions in radian

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.16 DecomposeHomography

Decompose a homography matrix and verify four solutions of rotation, translation, and plane normal. It uses visible reference points being in front of the camera to confirm which solution(s) of the four solutions (maximum two solutions) are consistent with all reference points.

Syntax

Definition:

```
HRESULT DecomposeHomography(
    HRESULT              hrPrev,
    TcVnMatrix3x3_LREAL& aHomography,
    TcVnMatrix3x3_LREAL& aCameraMatrix,
    TcVnMatrix3x3_LREAL& aRotationMatrixA,
    TcVnVector3_LREAL&   aTranslationVectorA,
    TcVnVector3_LREAL&   aNormVectorA,
    TcVnMatrix3x3_LREAL& aRotationMatrixB,
    TcVnVector3_LREAL&   aTranslationVectorB,
    TcVnVector3_LREAL&   aNormVectorB,
    ULONG&               nSolutions,
    ITcVnContainer*      ipPointsSrc = nullptr,
```

```
    ITcVnContainer*      ipPointsDes = nullptr,
    ITcVnContainer*      ipInlierMask = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aHomography	TcVnMatrix3x3_LREAL [▶ 1713]&		Homography matrix (a perspective transformation between two planes)
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&		Camera matrix
aRotationMatrixA	TcVnMatrix3x3_LREAL [▶ 1713]&		Rotation matrix of the first solution (Ra)
aTranslationVectorA	TcVnVector3_LREAL [▶ 1713]&		Translation vector of the first solution (Ta)
aNormVectorA	TcVnVector3_LREAL [▶ 1713]&		Norm vector of the first solution (Na)
aRotationMatrixB	TcVnMatrix3x3_LREAL [▶ 1713]&		Rotation matrix of the second solution (Rb)
aTranslationVectorB	TcVnVector3_LREAL [▶ 1713]&		Translation vector of the second solution (Tb)
aNormVectorB	TcVnVector3_LREAL [▶ 1713]&		Norm vector of the second solution (Nb)
nSolutions	ULONG&		Return the number of true potential solutions [0,1, or 2]. 0: no true potential solutions for the given corresponding points are confirmed. 1: the solution (Ra, Ta, Na) is confirmed. 2: both solution (Ra, Ta, Na) and (Rb, Tb, Nb) are confirmed. In all cases the function returns two solutions (Ra, Ta, Nb) and (Rb, Tb, Nb). The third and the fourth solutions can be calculated as follows: (Ra, -Ta, -Na) and (Rb, -Tb, -Nb).
ipPointsSrc	ITcVnContainer* [▶ 1890]	nullptr	Container with source points (ContainerType_Vector_TcVnPoint2_REAL)
ipPointsDes	ITcVnContainer* [▶ 1890]	nullptr	Container with destination points (same number as ipPoints1, ContainerType_Vector_TcVnPoint2_REAL)

Name	Type	Default	Description
ipInlierMask	ITcVnContainer* [▶ 1890]	nullptr	A mask marking the inliers (optional, set to 0 if not available; ContainerType_Vector_SINT or ContainerType_Vector_USINT)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.17 FlipImage

Flip an image (mirror and shift back to prior coordinates).

Syntax

Definition:

```
HRESULT FlipImage(
    HRESULT       hrPrev,
    ITcVnImage*   ipSrcImage,
    ITcVnImage*&  ipDestImage,
    ETcVnFlipAxis eFlipAxis
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
eFlipAxis	ETcVnFlipAxis [▶ 1742]	Selects the axis around which to flip the image

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.18 GenerateAffineTransformationUnitMatrix2D

Generate an affine transformation 2D unit matrix.

Syntax

Definition:

```
HRESULT GenerateAffineTransformationUnitMatrix2D(
    HRESULT              hrPrev,
    TcVnMatrix2x3_LREAL& aTransformationMatrix
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1713]&	Resulting affine transformation unit matrix

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.19 GetAffineTransformation

Calculate the affine transformation between three corresponding point pairs. The points mark the corners of the corresponding triangles.

Syntax

Definition:

```
HRESULT GetAffineTransformation(
    HRESULT                 hrPrev,
    TcVnArray3_Point2_REAL& aSrcPoints,
    TcVnArray3_Point2_REAL& aDestPoints,
    TcVnMatrix2x3_LREAL&    aTransformationMatrix
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aSrcPoints	TcVnArray3_Point2_REAL [▶ 1713]&	Source points
aDestPoints	TcVnArray3_Point2_REAL [▶ 1713]&	Destination points
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1713]&	Resulting transformation matrix

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.20 GetAffineTransformation2D

Estimate the affine transformation between two planar point sets.

Syntax

Definition:

```
HRESULT GetAffineTransformation2D(
    HRESULT                 hrPrev,
    ITcVnContainer*         ipSrcPoints,
    ITcVnContainer*         ipDestPoints,
    TcVnMatrix2x3_LREAL&    aAffineTransform,
    ETcVnEstimationAlgorithm eAlgorithm = EA_RANSAC,
    double                  fReprojThreshold = 3,
    ITcVnContainer**        pipInlierMask = nullptr,
    ULONG                   nMaxIterations = 2000,
    double                  fConfidence = 0.995,
    ULONG                   nRefineIters = 10
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPoints	ITcVnContainer* [▶ 1890]		Container with at least 3 source points (ContainerType_Vector_TcVnPoint2_REAL)
ipDestPoints	ITcVnContainer* [▶ 1890]		Container with destination points (same number as ipSrcPoints, ContainerType_Vector_TcVnPoint2_REAL)
aAffineTransform	TcVnMatrix2x3_LREAL [▶ 1713]&		Returns the affine transformation matrix, which transforms the source points to the destination points
eAlgorithm	ETcVnEstimationAlgorithm [▶ 1739]	EA_RANSAC	Estimation algorithm (only RANSAC and LMEDS are supported)
fReprojThreshold	double	3	Maximum allowed reprojection error to treat a point pair as an inlier
pipInlierMask	ITcVnContainer* [▶ 1890]*	nullptr	Returns a mask marking the inliers (optional, set to 0 if not required; ContainerType_Vector_USINT)
nMaxIterations	ULONG	2000	Maximum number of iterations
fConfidence	double	0.995	Confidence (0..1)
nRefineIters	ULONG	10	Maximum number of iterations of Levenberg-Marquardt algorithm to refine further the affine transform (using only inliers). Set to 0 to disable refining.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.21 GetPerspectiveTransformation

Calculate the perspective transformation between four corresponding point pairs. The points mark the corners of the corresponding rectangles.

Syntax

Definition:

```
HRESULT GetPerspectiveTransformation(
    HRESULT                 hrPrev,
    TcVnArray4_Point2_REAL& aSrcPoints,
    TcVnArray4_Point2_REAL& aDestPoints,
    TcVnMatrix3x3_LREAL&    aTransformationMatrix
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aSrcPoints	TcVnArray4_Point2_REAL [▶ 1713]&	Source points
aDestPoints	TcVnArray4_Point2_REAL [▶ 1713]&	Destination points
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	Resulting transformation matrix

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.22 Homography

Find the homography (perspective transformation) between two planar point sets.

Syntax

Definition:

```
HRESULT Homography(
    HRESULT                   hrPrev,
    ITcVnContainer*           ipSrcPoints,
    ITcVnContainer*           ipDestPoints,
    TcVnMatrix3x3_LREAL&      aPerspectiveTransform,
    ETcVnEstimationAlgorithm eAlgorithm = EA_DEFAULT,
    double                    fReprojThreshold = 3,
    ITcVnContainer**          pipInlierMask = nullptr,
    ULONG                     nMaxIterations = 2000,
    double                    fConfidence = 0.995
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPoints	ITcVnContainer* [▶ 1890]		Container with at least 4 source points (ContainerType_Vector_TcVnPoint2_REAL)
ipDestPoints	ITcVnContainer* [▶ 1890]		Container with destination points (same number as ipSrcPoints, ContainerType_Vector_TcVnPoint2_REAL)
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 1713]&		Returns the perspective transformation matrix, which transforms the source points to the destination points
eAlgorithm	ETcVnEstimationAlgorithm [▶ 1739]	EA_DEFAULT	Estimation algorithm
fReprojThreshold	double	3	Maximum allowed reprojection error to treat a point pair as an inlier (only for RANSAC, RHO)
pipInlierMask	ITcVnContainer* [▶ 1890]*	nullptr	Returns a mask marking the inliers (optional, set to 0 if not required; ContainerType_Vector_SINT; only for RANSAC, LMEDS)
nMaxIterations	ULONG	2000	Maximum number of RANSAC iterations
fConfidence	double	0.995	Confidence (0..1)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.23 InvertAffineTransformation

Invert a 2D affine transformation matrix. The inverted transformation compensates the affine transformation on an image if it is used in the F_VN_WarpAffine function.

Syntax

Definition:

```
HRESULT InvertAffineTransformation(
    HRESULT              hrPrev,
    TcVnMatrix2x3_LREAL& aTransformationMatrix,
    TcVnMatrix2x3_LREAL& aInvertedTransformationMatrix
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1713]&	Source affine transformation matrix
aInvertedTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1713]&	Resulting inverted affine transformation matrix

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.24 PerspectiveTransformation

Apply a perspective transformation (rotation + translation, e.g. from extrinsic calibration) to 3D point coordinates.

Syntax

Definition:

```
HRESULT PerspectiveTransformation(
    HRESULT              hrPrev,
    ITcVnContainer*      ipSrcPoints,
    ITcVnContainer*&     ipDestPoints,
    TcVnMatrix3x3_LREAL& aRotationMatrix,
    TcVnVector3_LREAL&   aTranslationVector
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPoints	ITcVnContainer* [▶ 1890]	Container with 3D source points (TcVnPoint3_REAL or TcVnPoint3_LREAL)
ipDestPoints	ITcVnContainer* [▶ 1890]&	Returns the transformed points (same type as ipSrcPoints)
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	3x3 rotation matrix
aTranslationVector	TcVnVector3_LREAL [▶ 1713]&	Translation vector

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.25 PyramidDown

Downsamples an image to half width and height.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT PyramidDown(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.26 PyramidUp

Upsamples an image to double width and height.

Syntax

Definition:

```
HRESULT PyramidUp(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.27 RemapImageToLogPolarSpace

Remap an image to log-polar space.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT RemapImageToLogPolarSpace(
    HRESULT               hrPrev,
    ITcVnImage*           ipSrcImage,
```

```
    ITcVnImage*&            ipDestImage,
    TcVnPoint2_REAL&        aCenter,
    double                  fScale = 0,
    ETcVnInterpolationType eInterpolationType = IT_BILINEAR,
    LONG                    nDestWidth = -1,
    LONG                    nDestHeight = -1
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (Must not be the same as ipSrcImage! An appropriate destination image will be created if required.)
aCenter	TcVnPoint2_REAL [▶ 1711]&		Center point for the transformation
fScale	double	0	Magnitude scale parameter, i.e. image width / ln(radius) (set to 0 for auto select)
eInterpolationType	ETcVnInterpolationType [▶ 1744]	IT_BILINEAR	Interpolation type
nDestWidth	LONG	-1	Destination image width (set to -1 for source image width, 0 to auto scale to transformed radius, > 0 for a user defined width)
nDestHeight	LONG	-1	destination image height (set to -1 for source image height, 0 to auto scale to transformed radius * PI, > 0 for a user defined height)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.28 RemapImageToPolarSpace

Remap an image to polar space.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT RemapImageToPolarSpace(
    HRESULT                hrPrev,
    ITcVnImage*            ipSrcImage,
    ITcVnImage*&           ipDestImage,
    TcVnPoint2_REAL&       aCenter,
    double                 fMaxRadius = 0,
    ETcVnInterpolationType eInterpolationType = IT_BILINEAR,
    LONG                   nDestWidth = -1,
    LONG                   nDestHeight = -1
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (Must not be the same as ipSrcImage! An appropriate destination image will be created if required.)
aCenter	TcVnPoint2_REAL [▶ 1711]&		Center point for the transformation
fMaxRadius	double	0	Maximum radius for the transformation (set to 0 for auto select)
eInterpolationType	ETcVnInterpolationType [▶ 1744]	IT_BILINEAR	Interpolation type
nDestWidth	LONG	-1	Destination image width (set to -1 for source image width, 0 to auto scale to transformed radius, > 0 for a user defined width)
nDestHeight	LONG	-1	destination image height (set to -1 for source image height, 0 to auto scale to transformed radius * PI, > 0 for a user defined height)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.29 ResizeImage

Resize an image using a specific interpolation type while maintaining its aspect ratio.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT ResizeImage(
    HRESULT                hrPrev,
    ITcVnImage*            ipSrcImage,
    ITcVnImage*&           ipDestImage,
    ULONG                  nWidth,
    ULONG                  nHeight,
    ETcVnInterpolationType eInterpolationType,
    ETcVnPaddingMode       ePaddingMode = PM_NONE,
    TcVnVector4_LREAL&     aBorderValue = {0, 0, 0, 0}
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (An appropriate image will be created if required.)
nWidth	ULONG		New width
nHeight	ULONG		New height
eInterpolationType	ETcVnInterpolationType [▶ 1744]		Interpolation type
ePaddingMode	ETcVnPaddingMode [▶ 1749]	PM_NONE	Image padding mode
aBorderValue	TcVnVector4_LREAL [▶ 1713]&	{0, 0, 0, 0}	Border value, if TCVN_PM_LETTERBOX is used

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.30 RotateImage

Rotate an image by 90, 180, or 270 degrees in clockwise direction.

Syntax

Definition:

```
HRESULT RotateImage(
    HRESULT            hrPrev,
    ITcVnImage*        ipSrcImage,
    ITcVnImage*&       ipDestImage,
    ETcVnRotationAngle eAngle
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate image will be created if required.)
eAngle	ETcVnRotationAngle [▶ 1752]	Rotation angle

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.31 RotateImage (Exp)

Rotate an image by an individual rotation angle in degrees.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT RotateImage(
    HRESULT                    hrPrev,
    ITcVnImage*                ipSrcImage,
    ITcVnImage*&               ipDestImage,
    double                     fAngleDeg,
```

```
    bool                            bAdjustDestSize,
    ETcVnInterpolationType          eInterpolationType = IT_BILINEAR,
    ETcVnBorderInterpolationMethod eBorderInterpolation = BIM_CONSTANT,
    TcVnVector4_LREAL&              aBorderValue = {0, 0, 0, 0}
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (An appropriate image will be created if required.)
fAngleDeg	double		Rotation angle in degree (positive: counter-clockwise, negative: clockwise)
bAdjustDestSize	bool		If true, ipDestImage size is adjusted so that the whole rotated ipSrcImage is contained. Otherwise, ipDestImage size is set to ipSrcImage size.
eInterpolationType	ETcVnInterpolationType [▶ 1744]	IT_BILINEAR	Interpolation method
eBorderInterpolation	ETcVnBorderInterpolationMethod [▶ 1716]	BIM_CONSTANT	Border interpolation method (ISOLATED not supported)
aBorderValue	TcVnVector4_LREAL [▶ 1713]&	{0, 0, 0, 0}	Border value, if CONSTANT is used

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.32 WarpAffine

Apply an affine transformation to an image.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT WarpAffine(
    HRESULT              hrPrev,
    ITcVnImage*          ipSrcImage,
    ITcVnImage*&         ipDestImage,
    TcVnMatrix2x3_LREAL& aTransformationMatrix
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate image will be created if required.)
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1713]&	Affine transformation matrix

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.33 WarpAffine (Container)

Apply an affine transformation to a container of 2D points.

Syntax

Definition:

```
HRESULT WarpAffine(
    HRESULT              hrPrev,
    ITcVnContainer*      ipSrcPoints,
    ITcVnContainer*&     ipDestPoints,
    TcVnMatrix2x3_LREAL& aTransformationMatrix
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPoints	ITcVnContainer* [▶ 1890]	Source points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestPoints	ITcVnContainer* [▶ 1890]&	Returns the transformed points (same type ID as ipSrcPoints; Non-zero interface pointers are reused.)
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1713]&	Affine transformation matrix

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.34 WarpAffine (Point)

Apply an affine transformation to a 2D point.

Syntax

Definition:

```
HRESULT WarpAffine(
    HRESULT              hrPrev,
    TcVnPoint2_LREAL&    aSrcPoint,
    TcVnPoint2_LREAL&    aDestPoint,
    TcVnMatrix2x3_LREAL& aTransformationMatrix
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aSrcPoint	TcVnPoint2_LREAL [▶ 1711]&	Source point
aDestPoint	TcVnPoint2_LREAL [▶ 1711]&	Destination point
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1713]&	Affine transformation matrix

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.35 WarpAffine (Rectangle)

Use an affine transform to warp a rectangle.

Syntax

Definition:

```
HRESULT WarpAffine(
    HRESULT               hrPrev,
    ULONG                 nTopLeftX,
    ULONG                 nTopLeftY,
    ULONG                 nBottomRightX,
    ULONG                 nBottomRightY,
    ITcVnContainer*&      ipDestPoints,
    TcVnMatrix2x3_LREAL&  aTransformationMatrix
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nTopLeftX	ULONG	x coordinate of the top left corner
nTopLeftY	ULONG	y coordinate of the top left corner
nBottomRightX	ULONG	x coordinate of the bottom right corner
nBottomRightY	ULONG	y coordinate of the bottom right corner
ipDestPoints	ITcVnContainer* [▶ 1890]&	Returns the 4 transformed points (ContainerType_Vector_TcVnPoint2_REAL)
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1713]&	Affine transformation matrix

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.36 WarpAffine (Exp)

Apply an affine transformation to an image.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT WarpAffine(
    HRESULT                         hrPrev,
    ITcVnImage*                     ipSrcImage,
    ITcVnImage*&                    ipDestImage,
    TcVnMatrix2x3_LREAL&            aTransformationMatrix,
    ULONG                           nDestWidth,
    ULONG                           nDestHeight,
    ETcVnInterpolationType          eInterpolationType,
    bool                            bWarpInverse,
    ETcVnBorderInterpolationMethod  eBorderInterpolation,
    double                          fBorderValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate image will be created if required.)
aTransformationMatrix	TcVnMatrix2x3_LREAL [▶ 1713]&	Affine transformation matrix
nDestWidth	ULONG	Width of the destination image
nDestHeight	ULONG	Height of the destination image
eInterpolationType	ETcVnInterpolationType [▶ 1744]	Interpolation method
bWarpInverse	bool	Handle aTransformationMatrix as the inverse transformation
eBorderInterpolation	ETcVnBorderInterpolationMethod [▶ 1716]	Border interpolation method (ISOLATED not supported)
fBorderValue	double	Border value, if CONSTANT is used

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.37 WarpPerspective

Apply a perspective transformation to an image.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT WarpPerspective(
    HRESULT              hrPrev,
    ITcVnImage*          ipSrcImage,
    ITcVnImage*&         ipDestImage,
    TcVnMatrix3x3_LREAL& aTransformationMatrix
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate image will be created if required.)
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	Perspective transformation matrix

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.38 WarpPerspective (Container)

Apply a perspective transformation to a container of 2D points.

Syntax

Definition:

```
HRESULT WarpPerspective(
    HRESULT              hrPrev,
    ITcVnContainer*      ipSrcPoints,
```

```
    ITcVnContainer*&     ipDestPoints,
    TcVnMatrix3x3_LREAL& aTransformationMatrix
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcPoints	ITcVnContainer* [▶ 1890]	Source points (ContainerType_Vector_TcVnPoint2_DINT or ContainerType_Vector_TcVnPoint2_REAL or ContainerType_Vector_TcVnPoint2_LREAL)
ipDestPoints	ITcVnContainer* [▶ 1890]&	Returns the transformed points (same type ID as ipSrcPoints; Non-zero interface pointers are reused.)
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	Perspective transformation matrix

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.39 WarpPerspective (Point)

Apply a perspective transformation to a 2D point.

Syntax

Definition:

```
HRESULT WarpPerspective(
    HRESULT              hrPrev,
    TcVnPoint2_LREAL&    aSrcPoint,
    TcVnPoint2_LREAL&    aDestPoint,
    TcVnMatrix3x3_LREAL& aTransformationMatrix
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aSrcPoint	TcVnPoint2_LREAL [▶ 1711]&	Source point
aDestPoint	TcVnPoint2_LREAL [▶ 1711]&	Destination point
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	Perspective transformation matrix

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.40 WarpPerspective (Rectangle)

Use a perspective transform to warp a rectangle.

Syntax

Definition:

```
HRESULT WarpPerspective(
    HRESULT              hrPrev,
    ULONG                nTopLeftX,
    ULONG                nTopLeftY,
    ULONG                nBottomRightX,
    ULONG                nBottomRightY,
    ITcVnContainer*&     ipDestPoints,
    TcVnMatrix3x3_LREAL& aTransformationMatrix
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nTopLeftX	ULONG	x coordinate of the top left corner
nTopLeftY	ULONG	y coordinate of the top left corner
nBottomRightX	ULONG	x coordinate of the bottom right corner
nBottomRightY	ULONG	y coordinate of the bottom right corner
ipDestPoints	ITcVnContainer* [▶ 1890]&	Returns the 4 transformed points (ContainerType_Vector_TcVnPoint2_REAL)
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	Perspective transformation matrix

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.13.41 WarpPerspective (Exp)

Apply a perspective transformation to an image.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT WarpPerspective(
    HRESULT                         hrPrev,
    ITcVnImage*                     ipSrcImage,
    ITcVnImage*&                    ipDestImage,
    TcVnMatrix3x3_LREAL&            aTransformationMatrix,
    ULONG                           nDestWidth,
    ULONG                           nDestHeight,
    ETcVnInterpolationType          eInterpolationType,
    bool                            bWarpInverse,
    ETcVnBorderInterpolationMethod eBorderInterpolation,
    double                          fBorderValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate image will be created if required.)
aTransformationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	Perspective transformation matrix
nDestWidth	ULONG	Width of the destination image
nDestHeight	ULONG	Height of the destination image
eInterpolationType	ETcVnInterpolationType [▶ 1744]	Interpolation method
bWarpInverse	bool	Handle aTransformationMatrix as the inverse transformation
eBorderInterpolation	ETcVnBorderInterpolationMethod [▶ 1716]	Border interpolation method (ISOLATED not supported)
fBorderValue	double	Border value, if CONSTANT is used

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14 Image Analysis

Diese Gruppe enthält Funktionen zur Extraktion von Informationen aus Bildern.

Funktionen

Objektdetektion [▶ 2498]

- Kontursuche
- Objektfindung durch Hough Transformation
- Matching durch Bildvergleich

Statistische Bildmerkmale

- ImageAverage [▶ 2518]
- ImageAverageStdDev [▶ 2519]
- ImageAbsoluteAverage [▶ 2526]
- ImageMedian [▶ 2521]
- MaxPixelValue [▶ 2523]
- MinPixelValue [▶ 2524]

Kantendetektion

- CannyEdgeDetection [▶ 2512]

Finden zusammenhängender Bildregionen

- ConnectedComponents [▶ 2513]
- ConnectedComponentsWithStats [▶ 2514]
- GetConnectedComponent [▶ 2517]

Ausrichtung einer Region

- RegionOrientation [▶ 2525]
- UprightInnerRectangle [▶ 2527]

Sonstiges

- CountNonZeroPixels [▶ 2516]
- DistanceTransformation [▶ 2516]
- ImageCenterOfMass [▶ 2520]
- ImageMoments [▶ 2522]

6.2.3.14.1 Object Detection

Diese Untergruppe enthält Funktionen zur Objekterkennung.

Funktionen

Objektdetektion

- DetectBlobs [▶ 2499]
- FindContours [▶ 2501]
- FindContourHierarchyExp [▶ 2500]
- HoughCircles [▶ 2503]
- HoughLines [▶ 2505]
- HoughLinesP [▶ 2506]

Matching

- MatchImageHuMoments [▶ 2508]
- MatchTemplate [▶ 2508]
- MatchTemplateAndEvaluate [▶ 2510]

Sonstiges

- AdjustActiveContour [▶ 2498]

6.2.3.14.1.1 AdjustActiveContour

Adjust active contour (snake) in order to minimize its cummulative (internal and external) energy.

Syntax

Definition:

```
HRESULT AdjustActiveContour(
    HRESULT         hrPrev,
    ITcVnImage*     ipImage,
    ITcVnContainer* ipActiveContour,
    float           fAlpha,
    float           fBeta,
    float           fGamma,
    ULONG           nWindowWidth,
    ULONG           nWindowHeight,
    ULONG           nMaxIterations,
    bool            bUseGradient
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]	Source image (1 channel, USINT elements)
ipActiveContour	ITcVnContainer* [▶ 1890]	Initial contour, which will be adjusted
fAlpha	float	Continuity energy coefficient
fBeta	float	Curvature energy coefficient
fGamma	float	Image energy coefficient
nWindowWidth	ULONG	Window width (3, 5, 7, ...)
nWindowHeight	ULONG	Window height (3, 5, 7, ...)
nMaxIterations	ULONG	Maximum iterations
bUseGradient	bool	If true, the gradient magnitude is used as image energy (otherwise: pixel intensity)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.1.2 DetectBlobs

Detects blob-contours. Applies a threshold, a contour search and offers several options for filtering the found contours. Provides easy setup for multiple thresholds and combination of results.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT DetectBlobs(
    HRESULT                 hrPrev,
    ITcVnImage*             ipSrcImage,
    ITcVnContainer*&        ipBlobContours,
    TcVnParamsBlobDetection& stParams,
    TcVnPoint&              aOffset = {0, 0}
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (USINT or REAL, 1 channel or 3 channel (3 channel input is expected to be RGB and internally converted to Gray))
ipBlobContours	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the found contours (ContainerType_Vector_Vector_TcVnPoint2_DINT; The elements of this container are single contours. Non-zero interface pointers are reused.)
stParams	TcVnParamsBlobDetection [▶ 1771]&		Parameters to filter the detected contours.
aOffset	TcVnPoint [▶ 1711]&	{0, 0}	Offset by which every contour point is shifted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.1.3 FindContourHierarchy (Exp)

Search for object contours in a binary image and determine their hierarchical relationship.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT FindContourHierarchy(
    HRESULT                     hrPrev,
    ITcVnImage*                 ipSrcImage,
    ITcVnContainer*&            ipContours,
    ITcVnContainer*&            ipHierarchy,
    ETcVnContourRetrievalMode   eRetrievalMode,
```

```
    ETcVnContourApproximationMethod eApproximationMethod,
    TcVnPoint&                      aOffset
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (1 channel, binary)
ipContours	ITcVnContainer* [▶ 1890]&	Returns a container which is filled with the found contours (ContainerType_Vector_Vector_TcVnPoint2_DINT; The elements of this container are single contours. Non-zero interface pointers are reused.)
ipHierarchy	ITcVnContainer* [▶ 1890]&	Returns a container which is filled with information on the found contours' hierarchy (ContainerType_Vector_TcVnVector4_DINT; The elements of this container are four-dimensional vectors containing the 0-based indices of the next [0] and the previous contour [1] at the same level, the first child [2], and the parent [3]. Non-zero interface pointers are reused.)
eRetrievalMode	ETcVnContourRetrievalMode [▶ 1734]	Specifies which contours are retrieved and how their relationship is encoded
eApproximationMethod	ETcVnContourApproximationMethod [▶ 1733]	Contour encoding
aOffset	TcVnPoint [▶ 1711]&	Offset by which every contour point is shifted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.1.4 FindContours

Search for object contours in a binary image.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT FindContours(
    HRESULT                          hrPrev,
    ITcVnImage*                      ipSrcImage,
    ITcVnContainer*&                 ipContours,
    ETcVnContourRetrievalMode        eRetrievalMode = CRM_EXTERNAL,
    ETcVnContourApproximationMethod eApproximationMethod = CAM_SIMPLE,
    TcVnPoint&                       aOffset = {0, 0}
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (1 channel, binary)
ipContours	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the found contours (ContainerType_Vector_Vector_TcVnPoint2_DINT; The elements of this container are single contours. Non-zero interface pointers are reused.)
eRetrievalMode	ETcVnContourRetrievalMode [▶ 1734]	CRM_EXTERNAL	Specifies which contours are retrieved and how their relationship is encoded
eApproximationMethod	ETcVnContourApproximationMethod [▶ 1733]	CAM_SIMPLE	Contour encoding
aOffset	TcVnPoint [▶ 1711]&	{0, 0}	Offset by which every contour point is shifted

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.1.5 HoughCircles

Search for circles using the Hough transform.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT HoughCircles(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage,
    ITcVnContainer*& ipCircles,
    double           fInvAccuRatio,
    double           fMinDist,
    ETcVnHoughMethod eHoughMethod = HM_GRADIENT,
    double           fParam1 = 100,
    double           fParam2 = 100,
    ULONG            nMinRadius = 0,
    ULONG            nMaxRadius = 0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (USINT, 1 channel, gray-level)
ipCircles	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the found circles (ContainerType_Vector_TcVnVector3_REAL; Each container element contains the x coordinate of the circle center [0], the y coordinate of the circle center [1], and the radius [2], respectively. Non-zero interface pointers are reused.)
fInvAccuRatio	double		Inverted ratio of the accumulator size in relation to the source image's size (must be > 0. A value of 2 means that the size is halved in both directions.)
fMinDist	double		Smallest allowed distance of two circles (must be > 0)
eHoughMethod	ETcVnHoughMethod [▶ 1743]	HM_GRADIENT	Hough method to use (GRADIENT or GRADIENT_ALT)
fParam1	double	100	First method specific parameter (GRADIENT, GRADIENT_ALT: upper threshold for canny edge detection, which must be > 0)
fParam2	double	100	Second method specific parameter (GRADIENT: accumulator threshold for detecting circle centers, which must be > 0. GRADIENT_ALT: required circle perfectness, which must be > 0 and < 1, 1 would be a perfect circle)
nMinRadius	ULONG	0	Minimum circle radius allowed
nMaxRadius	ULONG	0	Maximum circle radius allowed (if 0, the value is internally set to max(image rows, image columns))

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.1.6 HoughLines

Search for lines using the standard Hough transform.

Syntax

Definition:

```
HRESULT HoughLines(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage,
    ITcVnContainer*& ipLines,
    double           fDistRes,
    double           fAngleRes,
    ULONG            nAccuThreshold,
    double           fDistResDiv = 0,
    double           fAngleResDiv = 0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (USINT, 1 channel, binary, may be modified by the function)
ipLines	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the found lines (ContainerType_Vector_TcVnVector2_REAL; Each container element contains the distance from the origin [0] in pixels and the rotation angle [1] in radians, respectively. Non-zero interface pointers are reused.)
fDistRes	double		Distance resolution of the accumulator (in pixels, must be > 0)
fAngleRes	double		Angle resolution of the accumulator (in radians, must be > 0)
nAccuThreshold	ULONG		Accumulator threshold
fDistResDiv	double	0	Divisor of the distance resolution for the multi-scale Hough transform (>= 0, must be > 0 if fAngleResDiv is > 0)
fAngleResDiv	double	0	Divisor of the angle resolution for the multi-scale Hough transform (>= 0, must be > 0 if fDistResDiv is > 0)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.1.7 HoughLinesP

Search for line segments using the probabilistic Hough transform.

Syntax

Definition:

```
HRESULT HoughLinesP(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage,
    ITcVnContainer*& ipLines,
    double           fDistRes,
    double           fAngleRes,
    ULONG            nAccuThreshold,
    double           fMinLineLength = 0,
    double           fMaxLineGap = 0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (USINT, 1 channel, binary, may be modified by the function)
ipLines	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the found line segments (ContainerType_Vector_TcVnVector4_DINT; Each container element contains the two ending points. Non-zero interface pointers are reused.)
fDistRes	double		Distance resolution of the accumulator (in pixels, must be > 0)
fAngleRes	double		Angle resolution of the accumulator (in radians, must be > 0)
nAccuThreshold	ULONG		Accumulator threshold
fMinLineLength	double	0	Minimum line length to search for
fMaxLineGap	double	0	Maximum gap between points on the same line

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.1.8 MatchImageHuMoments

Compare two images using the Hu moment invariants.

Syntax

Definition:

```
HRESULT MatchImageHuMoments(
    HRESULT                           hrPrev,
    ITcVnImage*                       ipImage1,
    ITcVnImage*                       ipImage2,
    ETcVnContoursMatchComparisonMethod eComparisonMethod,
    double&                           fDissimilarity
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage1	ITcVnImage* [▶ 1925]	First image (1 channel)
ipImage2	ITcVnImage* [▶ 1925]	Second image (1 channel)
eComparisonMethod	ETcVnContoursMatchComparisonMethod [▶ 1734]	Method used for comparing the Hu moment invariants of the images
fDissimilarity	double&	Returns the dissimilarity of the image Hu moment invariants depending on the chosen comparison method

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.1.9 MatchTemplate

Match a template image with every location in the source image and save the comparison results.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT MatchTemplate(
    HRESULT                   hrPrev,
    ITcVnImage*               ipSrcImage,
    ITcVnImage*               ipTemplateImage,
    ITcVnImage*&              ipResultImage,
```

```
    ETcVnTemplateMatchMethod eMatchMethod = TMM_CCORR_NORMED,
    ITcVnImage*              ipTemplateMask = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (USINT or REAL, 1 or 3 channels)
ipTemplateImage	ITcVnImage* [▶ 1925]		Template image (same type as ipSrcImage, smaller width and height)
ipResultImage	ITcVnImage* [▶ 1925]&		Returns the result image (REAL, 1 channel, dimensions: (ipSrcImage.width - ipTemplateImage.width + 1) x (ipSrcImage.height - ipTemplateImage.height + 1). The best match is the global minimum (SQDIFF(_NORMED)) or maximum (CCORR(_NORMED), CCOEFF(_NORMED)). The position in ipResultImage is the top-left corner of ipTemplateImage position in ipSrcImage)
eMatchMethod	ETcVnTemplateMatchMethod [▶ 1756]	TMM_CCORR_NORMED	Specifies the template match method
ipTemplateMask	ITcVnImage* [▶ 1925]	nullptr	Optional mask for ipTemplateImage (same type and size as ipTemplateImage)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.1.10 MatchTemplateAndEvaluate

Match a template image with every location in the source image and evaluate the comparison results.
Returns a sorted list of possible matches (best match first).
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT MatchTemplateAndEvaluate(
    HRESULT                  hrPrev,
    ITcVnImage*              ipSrcImage,
    ITcVnImage*              ipTemplateImage,
    ITcVnContainer*&         ipMatches,
    float                    fMatchThreshold,
    ETcVnTemplateMatchMethod eMatchMethod = TMM_CCORR_NORMED,
    ITcVnImage*              ipTemplateMask = nullptr,
    float                    fScaleFactor = 1,
    ETcVnInterpolationType   eInterpolationType = IT_BILINEAR,
    ITcVnContainer**         pipMatchValues = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (USINT or REAL, 1 or 3 channels)
ipTemplateImage	ITcVnImage* [▶ 1925]		Template image (same type as ipSrcImage, smaller width and height)
ipMatches	ITcVnContainer* [▶ 1890]&		Returns the matching positions (ContainerType_Vector_TcVnPoint2_DINT, where each element represents the top-left corner of ipTemplateImage) in ipSrcImage, sorted by relevance (best match first)
fMatchThreshold	float		Threshold to separate relevant from irrelevant matches (0..1 for NORMED methods, otherwise dependent on template size and content. To find a suitable value, you could evaluate some sample result images of F_VN_MatchTemplateExp.)
eMatchMethod	ETcVnTemplateMatchMethod [▶ 1756]	TMM_CCORR_NORMED	Specifies the template match method
ipTemplateMask	ITcVnImage* [▶ 1925]	nullptr	Optional mask for ipTemplateImage (same type and size as ipTemplateImage)
fScaleFactor	float	1	Factor (0..1] to reduce source and template image width and height for better performance (but less accuracy!)
eInterpolationType	ETcVnInterpolationType [▶ 1744]	IT_BILINEAR	Image resize interpolation type (only used if fScaleFactor != 1, TCVN_IT_BILINEAR recommended for most cases)
pipMatchValues	ITcVnContainer* [▶ 1890]*	nullptr	Optionally returns the matching values (ContainerType_Vector_REAL, same size and sort order as ipMatches. Set to 0 if not required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.2 CannyEdgeDetection

Find edges using the Canny edge detection algorithm.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT CannyEdgeDetection(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage,
    double       fThresholdLow,
    double       fThresholdHigh,
    ULONG        nApertureSize = 3,
    bool         bL2Gradient = false
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (elements of type USINT)
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (elements of type USINT. An appropriate destination image will be created if required.)
fThresholdLow	double		Low threshold
fThresholdHigh	double		High threshold
nApertureSize	ULONG	3	Aperture size for the Sobel operator (3, 5, 7)
bL2Gradient	bool	false	If true, the more accurate (and slower) L2 norm is used instead of the L1 norm

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.3 ConnectedComponents

Computes the connected components of an image.

Syntax

Definition:

```
HRESULT ConnectedComponents(
    HRESULT                           hrPrev,
    ITcVnImage*                       ipSrcImage,
    ITcVnImage*&                      ipLabelImage,
    LONG&                             nLabels,
    ETcVnPixelConnectivity            eConnectivity = PC_8,
    ETcVnElementType                  eLabelType = TCVN_ET_DINT,
    ETcVnConnectedComponentsAlgorithm eAlgorithm = CCA_GRANA
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (USINT, 1 channel, binary)
ipLabelImage	ITcVnImage* [▶ 1925]&		Returns the labels for each source image pixel (1 channel, type depends on eLabelType. An appropriate destination image will be created if required.)
nLabels	LONG&		Returns the number of labels
eConnectivity	ETcVnPixelConnectivity [▶ 1750]	PC_8	Selects if 4- or 8-way pixel connectivity should be used
eLabelType	ETcVnElementType [▶ 1738]	TCVN_ET_DINT	Selects the type of ipLabelImage (only UINT or DINT supported)
eAlgorithm	ETcVnConnectedComponentsAlgorithm [▶ 1732]	CCA_GRANA	Selects the applied algorithm

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.4 ConnectedComponentsWithStats

Computes the connected components and corresponding statistics of an image.

Syntax

Definition:

```
HRESULT ConnectedComponentsWithStats(
    HRESULT                           hrPrev,
    ITcVnImage*                       ipSrcImage,
    ITcVnImage*&                      ipLabelImage,
    ITcVnContainer*&                  ipBoundingBoxes,
    ITcVnContainer*&                  ipNumPixels,
    ITcVnContainer*&                  ipCentroids,
    LONG&                             nLabels,
    ETcVnPixelConnectivity            eConnectivity = PC_8,
    ETcVnElementType                  eLabelType = TCVN_ET_DINT,
    ETcVnConnectedComponentsAlgorithm eAlgorithm = CCA_GRANA
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (USINT, 1 channel, binary)
ipLabelImage	ITcVnImage* [▶ 1925]&		Returns the labels for each source image pixel (1 channel, DINT. An appropriate destination image will be created if required.)
ipBoundingBoxes	ITcVnContainer* [▶ 1890]&		Returns the bounding boxes for each labeled region. (ContainerType_Vector_TcVnRectangle_DINT; Non-zero interface pointers are reused.)
ipNumPixels	ITcVnContainer* [▶ 1890]&		Returns the number of pixels for each labeled region. (ContainerType_Vector_DINT; Non-zero interface pointers are reused.)
ipCentroids	ITcVnContainer* [▶ 1890]&		Returns the centroids for each labeled region. (ContainerType_Vector_TcVnPoint2_LREAL; Non-zero interface pointers are reused.)
nLabels	LONG&		Returns the number of labels
eConnectivity	ETcVnPixelConnectivity [▶ 1750]	PC_8	Selects if 4- or 8-way pixel connectivity should be used
eLabelType	ETcVnElementType [▶ 1738]	TCVN_ET_DINT	Selects the type of ipLabelImage (only UINT or DINT supported)
eAlgorithm	ETcVnConnectedComponentsAlgorithm [▶ 1732]	CCA_GRANA	Selects the applied algorithm

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.5 CountNonZeroPixels

Counts the non-zero pixels in a single-channel image (e.g. useful to analyze threshold results).

Syntax

Definition:

```
HRESULT CountNonZeroPixels(
    HRESULT     hrPrev,
    ITcVnImage* ipSrcImage,
    ULONGLONG&  nNonZero
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (1 channel)
nNonZero	ULONGLONG&	Returns the number of non-zero pixels in ipSrcImage

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.6 DistanceTransformation

Calculates the distance transformation, which is the distance to the closest zero pixel in a binary image. Additionally, an image with component labels is created (discrete Voronoi diagram).

Syntax

Definition:

```
HRESULT DistanceTransformation(
    HRESULT                          hrPrev,
    ITcVnImage*                      ipSrcImage,
    ITcVnImage*&                     ipDestImage,
    ETcVnDistanceType                eDistanceType,
    ETcVnDistanceTransformationMask  eMaskSize,
    ITcVnImage**                     pipDestLabels = nullptr,
    ETcVnDistanceTransformationLabel eLabelType = DTL_CCOMP
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (USINT, 1 channel)
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (REAL, 1 channel. An appropriate destination image will be created if required.)
eDistanceType	ETcVnDistanceType [▶ 1736]		Distance computation method (supported: L1, L2, C)
eMaskSize	ETcVnDistanceTransformationMask [▶ 1736]		Size of the distance transformation mask (PRECISE not supported for label computation)
pipDestLabels	ITcVnImage* [▶ 1925]*	nullptr	Returns the component labels (Discrete Voronoi diagram; DINT, 1 channel. An appropriate image will be created if required.)
eLabelType	ETcVnDistanceTransformationLabel [▶ 1735]	DTL_CCOMP	Type of the labels

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.7 GetConnectedComponent

Get the mask image (and optionally the outer contour) of a connected component from a label image based on the label ID.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT GetConnectedComponent(
    HRESULT          hrPrev,
    ITcVnImage*      ipLabelImage,
    LONG             nLabelID,
```

```
    ITcVnImage*&     ipMaskImage,
    ITcVnContainer** pipContourPoints = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipLabelImage	ITcVnImage* [▶ 1925]		Image containing labels for each pixel (1 channel DINT or UINT).
nLabelID	LONG		Value of the requested label
ipMaskImage	ITcVnImage* [▶ 1925]&		Returns a mask image marking the pixels that belong to the component (1 channel USINT. Mask is optional, set to 0 if not required.)
pipContourPoints	ITcVnContainer* [▶ 1890]*	nullptr	Returns the outer contour of the component(ContainerType_Vector_TcVnPoint2_DINT. Contour is optional, set to 0 if not required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.8 ImageAverage

Computes the (channel-wise) average pixel value of an image.

Syntax

Definition:

```
HRESULT ImageAverage(
    HRESULT            hrPrev,
    ITcVnImage*        ipSrcImage,
    TcVnVector4_LREAL& aAverage,
    ITcVnImage*        ipMask = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (1 - 4 channels)
aAverage	TcVnVector4_LREAL [▶ 1713]&		Returns the (channel-wise) average pixel value of ipSrcImage
ipMask	ITcVnImage* [▶ 1925]	nullptr	Mask of type USINT (1 channel)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.9 ImageAverageStdDev

Computes the (channel-wise) average pixel value and the corresponding standard deviation of an image.

Syntax

Definition:

```
HRESULT ImageAverageStdDev(
    HRESULT            hrPrev,
    ITcVnImage*        ipSrcImage,
    TcVnVector4_LREAL& aAverage,
    TcVnVector4_LREAL& aStdDev,
    ITcVnImage*        ipMask = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (1 - 4 channels)
aAverage	TcVnVector4_LREAL [▶ 1713]&		Returns the (channel-wise) average pixel value of ipSrcImage
aStdDev	TcVnVector4_LREAL [▶ 1713]&		Returns the (channel-wise) pixel value standard deviation of ipSrcImage
ipMask	ITcVnImage* [▶ 1925]	nullptr	Mask of type USINT (1 channel)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.10 ImageCenterOfMass

Computes the center of mass of an image.

Syntax

Definition:

```
HRESULT ImageCenterOfMass(
    HRESULT           hrPrev,
    ITcVnImage*       ipSrcImage,
    TcVnPoint2_LREAL& aCenterOfMass,
    ITcVnImage*       ipMask = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (1 channel)
aCenterOfMass	TcVnPoint2_LREAL [▶ 1711]&		Returns the center of mass of the image
ipMask	ITcVnImage* [▶ 1925]	nullptr	Optional mask (1 channel of type USINT, same width and height as ipSrcImage)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.11 ImageMedian

Computes the (approximated, channel-wise) median pixel value of an image.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT ImageMedian(
    HRESULT            hrPrev,
    ITcVnImage*        ipSrcImage,
    TcVnVector4_LREAL& aMedian,
    ITcVnImage*        ipMask = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (1 - 4 channels)
aMedian	TcVnVector4_LREAL [▶ 1713]&		Returns the (approximated, channel-wise) median pixel value of ipSrcImage
ipMask	ITcVnImage* [▶ 1925]	nullptr	Mask of type USINT (1 channel)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.12 ImageMoments

Computes the spatial moments, the central moments, and the central normalized moments of an image up to the third order.

Syntax

Definition:

```
HRESULT ImageMoments(
    HRESULT      hrPrev,
    ITcVnImage*  ipImage,
    TcVnMoments& stMoments,
    bool         bBinaryImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]	Source image (1 channel, elements of type USINT or REAL)
stMoments	TcVnMoments [▶ 1769]&	Returns a struct containing the moments
bBinaryImage	bool	If true, all non-zero pixels are treated as ones

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.13 MaxPixelValue

Finds the maximum pixel value in an image (1 - 4 channels supported).

Syntax

Definition:

```
HRESULT MaxPixelValue(
    HRESULT                  hrPrev,
    ITcVnImage*              ipImage,
    TcVnVector4_LREAL&       aMaxValue,
    TcVnPoint2_DINT&         aPosition,
    ITcVnImage*              ipMask = nullptr,
    ETcVnVectorCompareMethod eVectorCompareMethod = VCM_EUCLIDEAN
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]		Source image
aMaxValue	TcVnVector4_LREAL [▶ 1713]&		Returns the maximum pixel value
aPosition	TcVnPoint2_DINT [▶ 1711]&		Returns the first found position of aMaxValue (not supported for multi-channel images with ELEMENTWISE)
ipMask	ITcVnImage* [▶ 1925]	nullptr	Optional mask to specify which pixel positions are considered (USINT, set parameter to 0 if not required)
eVectorCompareMethod	ETcVnVectorCompareMethod [▶ 1758]	VCM_EUCLIDEAN	Select a vector compare method for multi-channel images

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.14 MinPixelValue

Finds the minimum pixel value in an image (1 - 4 channels supported).

Syntax

Definition:

```
HRESULT MinPixelValue(
    HRESULT                 hrPrev,
    ITcVnImage*             ipImage,
    TcVnVector4_LREAL&      aMinValue,
    TcVnPoint2_DINT&        aPosition,
    ITcVnImage*             ipMask = nullptr,
    ETcVnVectorCompareMethod eVectorCompareMethod = VCM_EUCLIDEAN
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]		Source image
aMinValue	TcVnVector4_LREAL [▶ 1713]&		Returns the minimum pixel value
aPosition	TcVnPoint2_DINT [▶ 1711]&		Returns the first found position of aMinValue (not supported for multi-channel images with ELEMENTWISE)
ipMask	ITcVnImage* [▶ 1925]	nullptr	Optional mask to specify which pixel positions are considered (USINT, set parameter to 0 if not required)
eVectorCompareMethod	ETcVnVectorCompareMethod [▶ 1758]	VCM_EUCLIDEAN	Select a vector compare method for multi-channel images

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.15 RegionOrientation

Calculate the orientation of a region based on a binary image that contains a non zero value for every pixel in that region.

Syntax

Definition:

```
HRESULT RegionOrientation(
    HRESULT                hrPrev,
    ITcVnImage*            ipSrcImage,
    TcVnRotatedRectangle&  stOrientation,
    ETcVnOrientationMethod eMethod = OM_PCA
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (1 channel, binary)
stOrientation	TcVnRotatedRectangle [▶ 1782]&		Resulting rotated rectangle, containig center of mass, lenghts of axes, and rotation angle of the region in clockwise direction.
eMethod	ETcVnOrientationMethod [▶ 1749]	OM_PCA	Method for calculating the orientation.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.16 ImageAbsoluteAverage

Computes the average absolute pixel value of an image.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT ImageAbsoluteAverage(
    HRESULT     hrPrev,
    ITcVnImage* ipSrcImage,
    double&     fAverage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
fAverage	double&	Returns the average absolute pixel value

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.14.17 UprightInnerRectangle

Computes the maximum area upright inner rectangle inside an area.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT UprightInnerRectangle(
    HRESULT             hrPrev,
    ITcVnImage*         ipSrcImage,
    TcVnRectangle_DINT& stRectangle,
    LONG                nDownScaleFactor = -1
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (USINT, 1 channel, binary, values > 0 mark valid area)
stRectangle	TcVnRectangle_DINT [▶ 1781]&		Returns the maximum area upright inner rectangle
nDownScaleFactor	LONG	-1	To tradeoff performance vs. accuracy, a factor can be provided, to reduce the internally used image resolution. As long as invalid areas enclosed in the valid area are larger than nDownScaleFactor, the result usually still is the optimal one. Use -1 to auto select depending on ipSrcImage resolution, 0 to always use full resolution (slower but always optimal result).

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.15 Image Color and Contrast Processing

Diese Gruppe enthält Funktionen zur Farb- und Kontrastverarbeitung.

Funktionen

Häufigkeitsverteilung, Intensität und Kontrast Anpassung eines Bildes

- Clahe [▶ 2530]
- Histogram [▶ 2534]
- HistogramEqualization [▶ 2535]
- NormalizeImage [▶ 2536]
- NormalizeImageForDisplay [▶ 2538]

Look-Up Table und Color Maps

- ApplyColorMap [▶ 2528]
- ApplyLut [▶ 2529]
- GenerateColorMap [▶ 2532]
- GenerateCustomColorMap [▶ 2533]

Color Matching

- ReferenceColorSimilarity [▶ 2538]
- TrainImageColor [▶ 2542]
- TrainImageColor_ITcVnMlModel [▶ 2545]

Weißabgleich

- WhiteBalance [▶ 2548]

Sonstiges

- ConvertColorSpace [▶ 2531]
- InvertImageColor [▶ 2536]

6.2.3.15.1 ApplyColorMap

Apply a color map to a gray-level image.

Syntax

Definition:

```
HRESULT ApplyColorMap(
    HRESULT         hrPrev,
    ITcVnImage*     ipSrcImage,
    ITcVnImage*&    ipDestImage,
    ITcVnContainer* ipColorMap
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (TCVN_ET_USINT or TCVN_ET_UINT, 1 channel)
ipDestImage	ITcVnImage* [▶ 1925]&	Returns the colored image (TCVN_ET_USINT, 3 channel. An appropriate image will be created if required.)
ipColorMap	ITcVnContainer* [▶ 1890]	Color map to be applied to ipSrcImage (ContainerType_Vector_TcVnVector3_REAL with 256 or 65536 elements, dependent on ipSrcImage type). Can be either custom or created with F_VN_GenerateColorMap.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.15.2 ApplyLut

Apply a lookup table to an image to manipulate its colors.

Syntax

Definition:

```
HRESULT ApplyLut(
    HRESULT         hrPrev,
    ITcVnImage*     ipSrcImage,
    ITcVnImage*&    ipDestImage,
    ITcVnContainer* ipLut
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (1-4 channels of 8 or 16 bit types SINT, USINT, INT, UINT)
ipDestImage	ITcVnImage* [▶ 1925]&	Returns the result image (same size, type and channels as ipSrcImage. An appropriate image will be created if required.)
ipLut	ITcVnContainer* [▶ 1890]	Lookup table with elements matching ipSrcImage type and either 256 (8 bit images) or 65536 (16 bit images) elements. E.g. ContainerType_Vector_USINT for 1-4 channel USINT image (in this case, the same lookup values are used for each channel) or ContainerType_Vector_TcVnVector3_USINT for a 3 channel USINT image (in this case, each channel uses an individual lookup value).

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.15.3 Clahe

Apply Constrast Limited Adaptive Histogram Equalization. This applies local histogram equalization to individual tiles.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT Clahe(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage,
    double       fClipLimit = 40,
    ULONG        nTilesX = 8,
    ULONG        nTilesY = 8
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (TCVN_ET_USINT or TCVN_ET_UINT, 1 channel)
ipDestImage	ITcVnImage* [▶ 1925]&		Returns the resulting image (An appropriate image will be created if required.)
fClipLimit	double	40	Threshold for contrast limit (set to <= 0 to disable clipping)
nTilesX	ULONG	8	Number of tiles in x direction
nTilesY	ULONG	8	Number of tiles in y direction

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.15.4 ConvertColorSpace

Convert image from one color space to another.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT ConvertColorSpace(
    HRESULT                 hrPrev,
    ITcVnImage*             ipSrcImage,
    ITcVnImage*&            ipDestImage,
    ETcVnColorSpaceTransform eTransform
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
eTransform	ETcVnColorSpaceTransform [▶ 1721]	Transforms to be applied

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.15.5 GenerateColorMap

Generates a pre-defined color map.

Syntax

Definition:

```
HRESULT GenerateColorMap(
    HRESULT           hrPrev,
    ITcVnContainer*&  ipColorMap,
    ETcVnColorMap     eColorMap,
    ETcVnColorMapSize eColorMapSize
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipColorMap	ITcVnContainer* [▶ 1890]&	Returns the color map (ContainerType_Vector_TcVnVector3_REAL with 256 or 65536 elements)
eColorMap	ETcVnColorMap [▶ 1719]	Selects a color map (similar to GNU Octave/MATLAB types)
eColorMapSize	ETcVnColorMapSize [▶ 1720]	Defines how many elements the generated color map should have

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.15.6 GenerateCustomColorMap

Generate a custom color map by interpolating between user defined colors (equally distributed, linear for each channel).

Syntax

Definition:

```
HRESULT GenerateCustomColorMap(
    HRESULT           hrPrev,
    ITcVnContainer*&  ipColorMap,
    ITcVnContainer*   ipInitialColors,
    ETcVnColorMapSize eColorMapSize
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipColorMap	ITcVnContainer* [▶ 1890]&	Returns the color map (ContainerType_Vector_TcVnVector3_REAL with 256 or 65536 elements)
ipInitialColors	ITcVnContainer* [▶ 1890]	User defined colors (ContainerType_Vector_TcVnVector3_REAL, at least 2 elements)
eColorMapSize	ETcVnColorMapSize [▶ 1720]	Defines how many elements the generated color map should have

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.15.7 Histogram

Calculate the (multi-channel) histogram of an image.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT Histogram(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage,
    ITcVnContainer*& ipDestHistogram,
    ULONG&           nBins = 0,
    double&          fLowerBound = 0,
    double&          fUpperBound = 0,
    ITcVnImage*      ipMask = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipDestHistogram	ITcVnContainer* [▶ 1890]&		Returns a container with a multi-channel histogram, where every channel is represented as a vector of UDINT (ContainerType_Vector_Vector_UDINT. Non-zero interface pointers are reused.)
nBins	ULONG&	0	Desired number of bins or 0 to keep the default for the corresponding image format (in) and default number of bins (out)
fLowerBound	double&	0	Lower (inclusive) boundary of the 0-th histogram bin (in), or receive the default if fLowerBound AND fUpperBound are set to 0 (out)
fUpperBound	double&	0	Upper (exclusive) boundary of the last histogram bin nBins-1 (in), or receive the default if fLowerBound AND fUpperBound are set to 0 (out)
ipMask	ITcVnImage* [▶ 1925]	nullptr	Source image mask (TCVN_ET_USINT, 1 channel. Mask is optional, set to 0 if not required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.15.8 HistogramEqualization

Equalize the histogram of a grayscale or color image, which normalizes the brightness and improves the contrast. The channel index that should be equalized has to be specified (-1 expects a RGB image, converts it to YCbCr, equalizes the Y channel and converts the image back to RGB).
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT HistogramEqualization(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage,
    SHORT        nChannelIdx = -1
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (USINT)
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (An appropriate destination image will be created if required.)
nChannelIdx	SHORT	-1	Index of the image channel that should be equalized (-1 expects a RGB image, converts it to YCbCr, equalizes the Y channel and converts the image back to RGB)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.15.9 InvertImageColor

Invert the color of an image. For signed integer or negative floating point values, only the signs will be switched. If the image contains only positive floating point values, each pixel value is subtracted from the maximum available pixel value (or 1.0, whatever is higher).

Syntax

Definition:

```
HRESULT InvertImageColor(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage,
    double       fMaxValue = -1
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (An appropriate image will be created if required.)
fMaxValue	double	-1	Maximum pixel value (e.g. if a 16 bit image contains 12 bit values). -1 means the default values are used.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.15.10 NormalizeImage

Normalize an image regarding its value range (e.g. stretch pixel values [50..150] to full range [0..255]) or scale the values regarding a specific normalization (e.g. L2-norm).

Syntax

Definition:

```
HRESULT NormalizeImage(
    HRESULT                hrPrev,
    ITcVnImage*            ipSrcImage,
    ITcVnImage*&           ipDestImage,
    double                 fAlpha = 0,
    double                 fBeta = 255,
    ETcVnNormalizationType eNormType = NT_MINMAX,
    ETcVnElementType       eDestType = TCVN_ET_SAME_AS_SOURCE,
    ITcVnImage*            ipMask = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (An appropriate destination image will be created if required.)
fAlpha	double	0	Lower range boundary (in case of normalizing the value range) or value to normalize to (|| ipDestImage|| = fAlpha)
fBeta	double	255	Upper range boundary (in case of normalizing the value range)
eNormType	ETcVnNormalizationType [▶ 1747]	NT_MINMAX	Normalization type (only INF, L1, L2 or MINMAX)
eDestType	ETcVnElementType [▶ 1738]	TCVN_ET_SAME_AS_SOURCE	Destination image depth (usually SAME_AS_SOURCE)
ipMask	ITcVnImage* [▶ 1925]	nullptr	Mask to restrict the normalization to specific pixel positions (set 0 to normalize the whole image)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.15.11 NormalizeImageForDisplay

Normalize an image for display, i.e. scale it to the full value range of the underlying data type (-1 to 1 for floating point).
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT NormalizeImageForDisplay(
    HRESULT                 hrPrev,
    ITcVnImage*             ipSrcImage,
    ITcVnImage*&            ipDestImage,
    ETcVnSignedNormalization eSignedNormalization
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
eSignedNormalization	ETcVnSignedNormalization [▶ 1753]	Option for normalizing signed values

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.15.12 ReferenceColorSimilarity (ITcVnColorModel*)

Computes the similarity to a reference color model for each pixel in the source image.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT ReferenceColorSimilarity(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage,
    ITcVnImage*&     ipDestImage,
    ITcVnColorModel* ipColorModel,
```

```
    float           fVariance = 0.15f,
    float           fLuminanceWeight = 0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		RGB source image (USINT, 3 channels)
ipDestImage	ITcVnImage* [▶ 1925]&		Returns the similarity to ipColorModel for each pixel in ipSrcImage (USINT, 1 channel. An appropriate destination image will be created if required.)
ipColorModel	ITcVnColorModel* [▶ 1931]		Color model
fVariance	float	0.15f	Allowed color variance (0.1 - 0.3 might be a good start to try)
fLuminanceWeight	float	0	Weight the impact of the luminance ([0..1], e.g. set to 0 to be more resistant to unequal illumination, but might be required to differentiate between some colors. Ignored if ipColorModel is of type RGB.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.15.13 ReferenceColorSimilarity (ITcVnMlModel*)

Computes the similarity to a reference color model for each pixel in the source image.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT ReferenceColorSimilarity(
    HRESULT       hrPrev,
    ITcVnImage*   ipSrcImage,
    ITcVnImage*&  ipDestImage,
    ITcVnMlModel* ipColorModel,
    float         fVariance = 0.15f,
    float         fLuminanceWeight = 0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		RGB source image (USINT, 3 channels)
ipDestImage	ITcVnImage* [▶ 1925]&		Returns the similarity to ipColorModel for each pixel in ipSrcImage (USINT, 1 channel. An appropriate destination image will be created if required.)
ipColorModel	ITcVnMlModel* [▶ 1933]		Color model
fVariance	float	0.15f	Allowed color variance (0.1 - 0.3 might be a good start to try)
fLuminanceWeight	float	0	Weight the impact of the luminance ([0..1], e.g. set to 0 to be more resistant to unequal illumination, but might be required to differentiate between some colors. Ignored if ipColorModel is of type RGB.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.15.14 ReferenceColorSimilarity (TcVnVector3_LREAL)

Computes the similarity to a reference color for each pixel in the source image.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT ReferenceColorSimilarity(
    HRESULT            hrPrev,
    ITcVnImage*        ipSrcImage,
    ITcVnImage*&       ipDestImage,
    TcVnVector3_LREAL& aRefColor,
    float              fVariance = 0.15f,
    float              fLuminanceWeight = 0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		RGB source image (USINT, 3 channels)
ipDestImage	ITcVnImage* [▶ 1925]&		Returns the similarity to aRefColor for each pixel in ipSrcImage (USINT, 1 channel. An appropriate destination image will be created if required.)
aRefColor	TcVnVector3_LREAL [▶ 1713]&		Reference color (RGB, [0..255])
fVariance	float	0.15f	Allowed color variance (0.1 - 0.3 might be a good start to try)
fLuminanceWeight	float	0	Weight the impact of the luminance ([0..1], e.g. set to 0 to be more resistant to unequal illumination, but might be required to differentiate between some colors)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.15.15 TrainImageColor

Create a new color model, describing the image color.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT TrainImageColor(
    HRESULT                  hrPrev,
    ITcVnImage*              ipSrcImage,
    ITcVnColorModel*&        ipColorModel,
    ULONG                    nDifferentColors,
    ETcVnColorTrainingMethod eMethod = CTM_LAB,
    ITcVnImage*              ipMask = nullptr,
    ULONG                    nSkipPixels = 0,
    ETcVnClusteringAlgorithm eClusteringAlgorithm = CA_KMEANSPP,
    LONG                     nMaxClusterRadius = 50,
    bool                     bSingleSplitSteps = true
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (3 channels (RGB) of type USINT)
ipColorModel	ITcVnColorModel* [▶ 1931]&		Returns the color model
nDifferentColors	ULONG		Maximum number of different colors to distinguish (if LBG is used as a clustering algorithm, the result might have less different colors, depending on nMaxClusterRadius)
eMethod	ETcVnColorTrainingMethod [▶ 1732]	CTM_LAB	Color training method
ipMask	ITcVnImage* [▶ 1925]	nullptr	Optional image mask (1 channel of type USINT, set to 0 if not required)
nSkipPixels	ULONG	0	Number of pixels to skip between each evaluated color sample (to achieve a better performance). 0 takes every pixel into account and tends to be more accurate.
eClusteringAlgorithm	ETcVnClusteringAlgorithm [▶ 1719]	CA_KMEANSPP	Clustering algorithm
nMaxClusterRadius	LONG	50	Only used for the LBG clustering algorithm. Maximum allowed radius (> 0) of a single cluster, i.e. clusters with a higher radius will be split into smaller ones, until a global number of nDifferentColors is reached.

Name	Type	Default	Description
bSingleSplitSteps	bool	true	Only used for the LBG clustering algorithm. If true, the global optimization is always run after a single cluster has been split. If false, several clusters are split within the same step before applying the global optimization. Applying the global optimization less often is faster, but can lead to less optimal results, especially having 2 nearby clusters that could be represented by 1)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.15.16 TrainImageColor (ITcVnMlModel*)

Create a new color model, describing the image color.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT TrainImageColor(
    HRESULT                  hrPrev,
    ITcVnImage*              ipSrcImage,
    ITcVnMlModel*&           ipColorModel,
    ULONG                    nDifferentColors,
    ETcVnColorTrainingMethod eMethod = CTM_LAB,
    ITcVnImage*              ipMask = nullptr,
    ULONG                    nSkipPixels = 0,
    ETcVnClusteringAlgorithm eClusteringAlgorithm = CA_KMEANSPP,
    LONG                     nMaxClusterRadius = 50,
    bool                     bSingleSplitSteps = true
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (3 channels (RGB) of type USINT)
ipColorModel	ITcVnMlModel* [▶ 1933]&		Returns the color model
nDifferentColors	ULONG		Maximum number of different colors to distinguish (if LBG is used as a clustering algorithm, the result might have less different colors, depending on nMaxClusterRadius)
eMethod	ETcVnColorTrainingMethod [▶ 1732]	CTM_LAB	Color training method
ipMask	ITcVnImage* [▶ 1925]	nullptr	Optional image mask (1 channel of type USINT, set to 0 if not required)
nSkipPixels	ULONG	0	Number of pixels to skip between each evaluated color sample (to achieve a better performance). 0 takes every pixel into account and tends to be more accurate.
eClusteringAlgorithm	ETcVnClusteringAlgorithm [▶ 1719]	CA_KMEANSPP	Clustering algorithm
nMaxClusterRadius	LONG	50	Only used for the LBG clustering algorithm. Maximum allowed radius (> 0) of a single cluster, i.e. clusters with a higher radius will be split into smaller ones, until a global number of nDifferentColors is reached.
bSingleSplitSteps	bool	true	Only used for the LBG clustering algorithm. If true, the global optimization is always run after a single cluster has been split. If false, several clusters are split within the same step before applying the global optimization. Applying the global optimization less often is faster, but can lead to less optimal results, especially having 2 nearby clusters that could be represented by 1)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.15.17 WhiteBalance

Calculates the white balance ratios for a RGB image. Uses the green channel as a reference and computes the relative ratios for the red and blue channels. Requires a non-reflecting, not overexposed white colored object (e.g. a sheet of paper) in at least 100 pixels of the image to provide reasonable results. The resulting ratios can then be set on the camera (multiply with existing ratios), or applied in the PLC via MultiplyImageWithVector.

Syntax

Definition:

```
HRESULT WhiteBalance(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage** pipDestImage,
    float&       fRatioRed,
    float&       fRatioBlue,
    ULONG        nMaxValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (3 channels (RGB), USINT or UINT)
pipDestImage	ITcVnImage* [▶ 1925]*	Destination image with white balance applied (optional, set to 0 if not required. An appropriate image will be created if required.)
fRatioRed	float&	Returns the ratio for the red channel
fRatioBlue	float&	Returns the ratio for the blue channel
nMaxValue	ULONG	Specify the maximum pixel value that would be achieved by overexposure (usually 255 for 8 bit or 4095 for 12 bit data)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.16 Image Filtering

Diese Gruppe enthält Funktionen zur Filterung von Bildern.

Funktionen

Weichzeichnung

- BilateralFilter [▶ 2550]
- BoxFilter [▶ 2551]
- GaussianFilter [▶ 2556]
- MedianFilter [▶ 2560]

Kantendetektion

- LaplacianFilter [▶ 2557]
- ScharrFilter [▶ 2563]
- SobelFilter [▶ 2565]

Benutzerdefinierte Filter

- CustomFilter [▶ 2553]
- SeparableCustomFilter [▶ 2564]

Flächen füllen

- BrightBorderObjects [▶ 2552]
- DarkBorderObjects [▶ 2554]
- FillHoles [▶ 2555]

Morphologischer Operator

- CreateStructuringElement [▶ 2552]
- MorphologicalOperator [▶ 2560]

Lokale Extrema

- LocalMaxima [▶ 2558]
- LocalMinima [▶ 2559]
- RemoveLocalMaxima [▶ 2561]
- RemoveLocalMinima [▶ 2562]

Lokale Varianz

- VarianceFilter [▶ 2566]

6.2.3.16.1 BilateralFilter

Apply a Bilateral filter to smooth the image but preserve edges.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT BilateralFilter(
    HRESULT                          hrPrev,
    ITcVnImage*                      ipSrcImage,
    ITcVnImage*&                     ipDestImage,
    LONG                             nDiameter,
    double                           fSigmaColor,
    double                           fSigmaSpace,
    ETcVnBorderInterpolationMethod eBorderType = BIM_DEFAULT
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (Must be different from ipSrcImage! An appropriate destination image will be created if required.)
nDiameter	LONG		Diameter of the pixel neighborhood used for filtering (e.g. 5, 7, 9). If <= 0, it is automatically chosen dependend on fSigmaSpace.
fSigmaColor	double		Sigma used for color space filtering (> 0). A larger value means that farther colors are mixed together.
fSigmaSpace	double		Sigma used for coordinate space filtering (> 0). A larger value means that farther pixels can influence each other.
eBorderType	ETcVnBorderInterpolationMethod [▶ 1716]	BIM_DEFAULT	Image border handling

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.16.2 BoxFilter

Apply a box filter to an image.

Syntax

Definition:

```
HRESULT BoxFilter(
    HRESULT                         hrPrev,
    ITcVnImage*                     ipSrcImage,
    ITcVnImage*&                    ipDestImage,
    ULONG                           nFilterWidth,
    ULONG                           nFilterHeight,
    ETcVnElementType                eDestDepth = TCVN_ET_SAME_AS_SOURCE,
    TcVnPoint&                      aAnchor = {-1, -1},
    bool                            bNormalize = true,
    ETcVnBorderInterpolationMethod eBorderType = BIM_DEFAULT
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (An appropriate destination image will be created if required.)
nFilterWidth	ULONG		Filter width in pixels
nFilterHeight	ULONG		Filter height in pixels
eDestDepth	ETcVnElementType [▶ 1738]	TCVN_ET_SAME_AS_SOURCE	Destination image depth
aAnchor	TcVnPoint [▶ 1711]&	{-1, -1}	Anchor point of the kernel ([-1, -1] for center)
bNormalize	bool	true	If true, the kernel is normalized by nFilterWidth * nFilterHeight
eBorderType	ETcVnBorderInterpolationMethod [▶ 1716]	BIM_DEFAULT	Image border handling

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.16.3 BrightBorderObjects

Find bright objects connected to the image border within a gray-scale single-channel image.

Syntax

Definition:

```
HRESULT BrightBorderObjects(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.16.4 CreateStructuringElement

Creates a structuring element for the usage with morphological operators and allocate an appropriate data buffer. The initial reference count is set to one if a new image interface is created and kept, otherwise.

Syntax

Definition:

```
HRESULT CreateStructuringElement(
    HRESULT                      hrPrev,
    ITcVnImage*&                 ipStructuringElement,
    ETcVnStructuringElementShape eShape,
    ULONG                        nWidth,
    ULONG                        nHeight
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipStructuringElement	ITcVnImage* [▶ 1925]&	Returns the created structuring element (Non-zero interface pointers are reused.)
eShape	ETcVnStructuringElementShape [▶ 1754]	Shape of the structuring element (rectangle, cross, or ellipse)
nWidth	ULONG	Width
nHeight	ULONG	Height

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.16.5 CustomFilter

Apply a custom filter to the image.

Syntax

Definition:

```
HRESULT CustomFilter(
    HRESULT                          hrPrev,
    ITcVnImage*                      ipSrcImage,
    ITcVnImage*&                     ipDestImage,
    ETcVnElementType                 eDestDepth,
    TcVnMatrix&                      stKernel,
    TcVnPoint&                       aAnchor = {-1, -1},
    double                           fDelta = 0,
    ETcVnBorderInterpolationMethod eBorderType = BIM_DEFAULT
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 1738]		Destination image depth
stKernel	TcVnMatrix [▶ 1769]&		Custom filter kernel with values of type REAL or LREAL
aAnchor	TcVnPoint [▶ 1711]&	{-1, -1}	Anchor point of the kernel
fDelta	double	0	Value added to each pixel after filtering
eBorderType	ETcVnBorderInterpolationMethod [▶ 1716]	BIM_DEFAULT	Image border handling

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.16.6 DarkBorderObjects

Find dark objects connected to the image border within a gray-scale single-channel image. (equivalent to filling holes)

Syntax

Definition:

```
HRESULT DarkBorderObjects(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.16.7 FillHoles

Fill holes within a gray-scale single-channel image. (equivalent to finding dark border objects)

Syntax

Definition:

```
HRESULT FillHoles(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.16.8 GaussianFilter

Apply a Gaussian filter to smooth the image.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT GaussianFilter(
    HRESULT                         hrPrev,
    ITcVnImage*                     ipSrcImage,
    ITcVnImage*&                    ipDestImage,
    ULONG                           nFilterWidth,
    ULONG                           nFilterHeight,
    double                          fSigmaX = 0,
    double                          fSigmaY = 0,
    ETcVnBorderInterpolationMethod eBorderType = BIM_DEFAULT
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (An appropriate destination image will be created if required.)
nFilterWidth	ULONG		Filter width in pixels (1, 3, 5, 7, ...)
nFilterHeight	ULONG		Filter height in pixels (1, 3, 5, 7, ...)
fSigmaX	double	0	Gaussian kernel standard deviation in X direction (>= 0, automatically chosen if 0)
fSigmaY	double	0	Gaussian kernel standard deviation in Y direction (>= 0, automatically chosen if 0)
eBorderType	ETcVnBorderInterpolationMethod [▶ 1716]	BIM_DEFAULT	Image border handling

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.16.9 LaplacianFilter

Apply a Laplacian filter to an image.

Syntax

Definition:

```
HRESULT LaplacianFilter(
    HRESULT                         hrPrev,
    ITcVnImage*                     ipSrcImage,
    ITcVnImage*&                    ipDestImage,
    ETcVnElementType                eDestDepth,
    ULONG                           nKernelSize = 1,
    double                          fScale = 1,
    double                          fDelta = 0,
    ETcVnBorderInterpolationMethod eBorderType = BIM_DEFAULT
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 1738]		Destination image depth
nKernelSize	ULONG	1	Aperture size used to compute the second-derivative filters (1, 3, 5, ..., 31)
fScale	double	1	Scale factor for the computed derivative values
fDelta	double	0	Delta value that is added to the results before storing them
eBorderType	ETcVnBorderInterpolationMethod [▶ 1716]	BIM_DEFAULT	Image border handling

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.16.10 LocalMaxima

Find local maxima in a gray-scale single-channel image. The found maxima are marked by a value of 1 in the destination image.

Syntax

Definition:

```
HRESULT LocalMaxima(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (USINT or UINT, 1 channel)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (Must be different from ipSrcImage! An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.16.11 LocalMinima

Find local minima in a gray-scale single-channel image. The found minima are marked by a value of 1 in the destination image.

Syntax

Definition:

```
HRESULT LocalMinima(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (USINT or UINT, 1 channel)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (Must be different from ipSrcImage! An appropriate destination image will be created if required.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.16.12 MedianFilter

Apply a Median filter to an image.

Syntax

Definition:

```
HRESULT MedianFilter(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage,
    ULONG        nFilterSize
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (for nFilterSize 3 or 5: USINT, UINT, REAL. For bigger filters, only USINT is supported.
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
nFilterSize	ULONG	Size (width and height) of the filter (3, 5, 7, ...)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.16.13 MorphologicalOperator

Apply a morphological operator.

Syntax

Definition:

```
HRESULT MorphologicalOperator(
    HRESULT                    hrPrev,
    ITcVnImage*                ipSrcImage,
    ITcVnImage*&               ipDestImage,
    ETcVnMorphologicalOperator eOperator,
    ITcVnImage*                ipStructuringElement
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (only 1 channel supported for reconstruction operators)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
eOperator	ETcVnMorphologicalOperator [▶ 1746]	Operator type
ipStructuringElement	ITcVnImage* [▶ 1925]	Structuring element to be applied (Typically created via F_VN_CreateStructuringElement.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.16.14 RemoveLocalMaxima

Remove local maxima up to a given height from a gray-scale single-channel image.

Syntax

Definition:

```
HRESULT RemoveLocalMaxima(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage,
    double       fHeight
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
fHeight	double	Maximum height of the maxima to be removed (must be greater than 0)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.16.15 RemoveLocalMinima

Remove local minima up to a given height from a gray-scale single-channel image.

Syntax

Definition:

```
HRESULT RemoveLocalMinima(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage,
    double       fHeight
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
fHeight	double	Maximum height of the minima to be removed (must be greater than 0)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.16.16 ScharrFilter

Calculates the first order derivative in x or y direction using a Scharr filter.

Syntax

Definition:

```
HRESULT ScharrFilter(
    HRESULT                         hrPrev,
    ITcVnImage*                     ipSrcImage,
    ITcVnImage*&                    ipDestImage,
    ETcVnElementType                eDestDepth,
    ETcVnFilterDirection            eFilterDirection,
    double                          fScale = 1,
    double                          fDelta = 0,
    ETcVnBorderInterpolationMethod eBorderType = BIM_DEFAULT
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 1738]		Destination image depth
eFilterDirection	ETcVnFilterDirection [▶ 1741]		Filter direction
fScale	double	1	Scale factor for the computed derivative values
fDelta	double	0	Delta value that is added to the results prior to storing them in dest
eBorderType	ETcVnBorderInterpolationMethod [▶ 1716]	BIM_DEFAULT	Image border handling

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.16.17 SeparableCustomFilter

Apply a separable custom filter to the image.

Syntax

Definition:

```
HRESULT SeparableCustomFilter(
    HRESULT                         hrPrev,
    ITcVnImage*                     ipSrcImage,
    ITcVnImage*&                    ipDestImage,
    ETcVnElementType                eDestDepth,
    TcVnMatrix&                     stKernelX,
    TcVnMatrix&                     stKernelY,
    TcVnPoint&                      aAnchor = {-1, -1},
    double                          fDelta = 0,
    ETcVnBorderInterpolationMethod eBorderType = BIM_DEFAULT
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 1738]		Destination image depth
stKernelX	TcVnMatrix [▶ 1769]&		1D custom row-filter kernel with values of type REAL or LREAL
stKernelY	TcVnMatrix [▶ 1769]&		1D custom column-filter kernel with values of type REAL or LREAL
aAnchor	TcVnPoint [▶ 1711]&	{-1, -1}	Anchor point of the kernel
fDelta	double	0	Value added to each pixel after filtering
eBorderType	ETcVnBorderInterpolationMethod [▶ 1716]	BIM_DEFAULT	Image border handling

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.16.18 SobelFilter

Calculates the first, second, third, or mixed image derivatives using an extended Sobel filter.

Syntax

Definition:

```
HRESULT SobelFilter(
    HRESULT                         hrPrev,
    ITcVnImage*                     ipSrcImage,
    ITcVnImage*&                    ipDestImage,
    ETcVnElementType                eDestDepth,
    ULONG                           nXOrder,
    ULONG                           nYOrder,
    ULONG                           nKernelSize = 3,
    double                          fScale = 1,
```

```
    double                          fDelta = 0,
    ETcVnBorderInterpolationMethod eBorderType = BIM_DEFAULT
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (An appropriate destination image will be created if required.)
eDestDepth	ETcVnElementType [▶ 1738]		Destination image depth
nXOrder	ULONG		Order of the x-derivative (must be < nKernelSize)
nYOrder	ULONG		Order of the y-derivative (must be < nKernelSize)
nKernelSize	ULONG	3	Size of the extended Sobel kernel (3, 5, 7, ..., 31)
fScale	double	1	Scale factor for the computed derivative values
fDelta	double	0	Delta value that is added to the results prior to storing them in dest
eBorderType	ETcVnBorderInterpolationMethod [▶ 1716]	BIM_DEFAULT	Image border handling

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.16.19 VarianceFilter

Calculates the local variance of an image.

Syntax

Definition:

```
HRESULT VarianceFilter(
    HRESULT                          hrPrev,
    ITcVnImage*                      ipSrcImage,
    ITcVnImage*&                     ipDestImage,
    ULONG                            nKernelSize,
    double                           fScale = 1,
    ETcVnBorderInterpolationMethod eBorderType = BIM_DEFAULT
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image. Only TCVN_ET_USINT Type is supported.
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image of type TCVN_ET_UINT (An appropriate destination image will be created if required.)
nKernelSize	ULONG		Size of the kernel
fScale	double	1	Scale factor for the computed variance values
eBorderType	ETcVnBorderInterpolationMethod [▶ 1716]	BIM_DEFAULT	Image border handling

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.17 Image Segmentation

bletDiese Gruppe enthält Funktionen zur Segmentierung von Bildern.

Funktionen

Segmentierung

- AdaptiveThreshold [▶ 2568]
- CheckColorRange [▶ 2570]
- DoubleThreshold [▶ 2570]
- Threshold [▶ 2571]

- WatershedSegmentationExp [▶ 2572]
- SauvolaThreshold [▶ 2573]

6.2.3.17.1 AdaptiveThreshold

Apply an adaptive threshold to a gray level image.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT AdaptiveThreshold(
    HRESULT                       hrPrev,
    ITcVnImage*                   ipSrcImage,
    ITcVnImage*&                  ipDestImage,
    double                        fMaxValue,
    ETcVnAdaptiveThresholdMethod  eAdaptiveMethod = ATM_MEAN,
    ETcVnThresholdType            eThresholdType = TT_BINARY,
    ULONG                         nBlockSize = 3,
    double                        fConstant = 5
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (1 channel, elements of type USINT)
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (An appropriate destination image will be created if required.)
fMaxValue	double		Value assigned to pixels for which the threshold condition is true
eAdaptiveMethod	ETcVnAdaptiveThresholdMethod [▶ 1714]	ATM_MEAN	Adaptive threshold method to be applied (MEAN: threshold value will be calculated as a mean of the nBlockSize x nBlockSize neighborhood of (x,y) minus fConstant. GAUSSIAN: threshold value is a weighted sum (cross-correlation with a Gaussian window) of the nBlockSize x nBlockSize neighborhood of (x,y) minus fConstant.
eThresholdType	ETcVnThresholdType [▶ 1757]	TT_BINARY	Threshold type to be applied (only BINARY and BINARY_INV are supported)
nBlockSize	ULONG	3	Size of the pixel neighborhood to calculate the local threshold (3, 5, 7, ...)
fConstant	double	5	Constant that is subtracted from the weighted mean of the pixel neighborhood, which leads to the local threshold

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.17.2 CheckColorRange

Check if the pixel values of an image lie in a given range. The destination image has the same size as the source image but only one channel with a pixel element size of 8 bit. Its elements are set to 255 if the corresponding pixels of the source image are in the checked range and set to 0 otherwise.

Syntax

Definition:

```
HRESULT CheckColorRange(
    HRESULT             hrPrev,
    ITcVnImage*         ipSrcImage,
    ITcVnImage*&        ipDestImage,
    TcVnVector4_LREAL&  aLowerBounds,
    TcVnVector4_LREAL&  aUpperBounds
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
aLowerBounds	TcVnVector4_LREAL [▶ 1713]&	Channel-wise lower bounds (Unused channels are ignored.)
aUpperBounds	TcVnVector4_LREAL [▶ 1713]&	Channel-wise upper bounds (Unused channels are ignored.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.17.3 DoubleThreshold

Apply a double threshold (also called hysteresis threshold) using morphological reconstruction.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT DoubleThreshold(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage,
    double       fLowThreshold,
    double       fHighThreshold,
    double       fMaxValue
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
fLowThreshold	double	Low threshold determinining the shape of found regions
fHighThreshold	double	High threshold selecting found regions (Each region must at least contain a single pixel if the high threshold is applied.)
fMaxValue	double	Maximum pixel value

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.17.4 Threshold

Apply a fixed threshold or a dynamic threshold according to Otsu.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT Threshold(
    HRESULT            hrPrev,
    ITcVnImage*        ipSrcImage,
    ITcVnImage*&       ipDestImage,
    double             fThreshold,
```

```
    double             fMaxValue,
    ETcVnThresholdType eThresholdType
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (For Otsu and Triangle threshold types, only 1 channel of TCVN_ET_USINT is supported)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
fThreshold	double	Fixed threshold (unused if dynamic thresholding is selected)
fMaxValue	double	Maximum pixel value
eThresholdType	ETcVnThresholdType [▶ 1757]	Threshold type to be applied

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.17.5 WatershedSegmentation (Exp)

Apply a marker-based watershed segmentation.

Syntax

Definition:

```
HRESULT WatershedSegmentation(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage,
    ITcVnImage*  ipMarkers
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (USINT, 3 channels)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (DINT, 1 channel. An appropriate image will be created if required.)
ipMarkers	ITcVnImage* [▶ 1925]	Marker image (DINT, 1 channel)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.17.6 SauvolaThreshold

Apply Sauvola local threshold to source image.

Syntax

Definition:

```
HRESULT SauvolaThreshold(
    HRESULT            hrPrev,
    ITcVnImage*        ipSrcImage,
    ITcVnImage*&       pipDestImage,
    ULONG              nBlockSize,
    double             fParamK,
    double             fMaxSigma,
    ETcVnThresholdType eThresholdType
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (USINT, UINT, INT, REAL, or LREAL, 1 channel)
pipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate single channel destination image will be created if required.))
nBlockSize	ULONG	Size of the pixel neighborhood to calculate the local threshold (3, 5, 7, ...)
fParamK	double	parameter K which takes positive values in range (0,1). Recommended are K in range [0.2, 0.5]
fMaxSigma	double	Maximum value of the standard deviation, required to be strictly positive. (fMaxSigma = 128 for a grayscale document).
eThresholdType	ETcVnThresholdType [▶ 1757]	Threshold type to be applied. Supported are TCVN_TT_BINARY and TCVN_TT_BINARY_INV.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.18 Keypoint Features

Diese Gruppe enthält Funktionen zum Finden und Matchen von markanten Features in einem Bild.

HINWEIS

Hohe Rechendauer

Viele Funktionen zum Finden von Keypoint Features benötigen eine hohe Rechendauer. Achten Sie daher auf die einstellte Zykluszeit und verwenden Sie ggfs. einen Watchdog [▶ 156] zur Absicherung des Echtzeitverhaltens.

Funktionen

Key Points

- KeyPointsAGAST [▶ 2593]
- KeyPointsFAST [▶ 2598]
- KeyPointsGFTT [▶ 2599]

- KeyPointsMSER [▶ 2600]
- KeyPointsSB [▶ 2601]

Key Points und Deskriptoren

- KeyPointsAndDescriptorsAKAZE [▶ 2594]
- KeyPointsAndDescriptorsBRISK [▶ 2595]
- KeyPointsAndDescriptorsaKAZE [▶ 2596]
- KeyPointsAndDescriptorsORB [▶ 2597]

Deskriptor-Matching

- FilterGoodMatches [▶ 2575]
- GetMatchCoordinates [▶ 2592]
- MatchDescriptorsBF [▶ 2602]
- MatchDescriptorsFlannLsh [▶ 2603]
- MatchDescriptorsKnnBF [▶ 2604]
- MatchDescriptorsKnnFlannLsh [▶ 2607]

Key Points Referenz Matching

- FindReferenceKeyPointsInImage [▶ 2576]
- FindReferenceKeyPointsInImageAKAZE [▶ 2580]
- FindReferenceKeyPointsInImageBRISK [▶ 2584]
- FindReferenceKeyPointsInImageORB [▶ 2588]

Sonstiges

- RegionsMSER [▶ 2608]

6.2.3.18.1 FilterGoodMatches

Filter the descriptor matches and return only good ones.

Syntax

Definition:

```
HRESULT FilterGoodMatches(
    HRESULT          hrPrev,
    ITcVnContainer*  ipMatches,
    ITcVnContainer*& ipGoodMatches,
    float            fMaxDist,
    float            fMaxKnnRatio
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMatches	ITcVnContainer* [▶ 1890]	Container with descriptor matches (ContainerType_Vector_TcVnDMatch or ContainerType_Vector_Vector_TcVnDMatch)
ipGoodMatches	ITcVnContainer* [▶ 1890]&	Returns a container with good matches (ContainerType_Vector_TcVnDMatch)
fMaxDist	float	Maximum allowed descriptor distance (-1 disables this filter criterion)
fMaxKnnRatio	float	Maximum allowed distance ratio between first and second best match ([0..1], used for knn match results only, -1 disables this filter criterion)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.18.2 FindReferenceKeyPointsInImage

Searches a reference image, represented by its keypoints and corresponding descriptors, in a given source image.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT FindReferenceKeyPointsInImage(
    HRESULT                 hrPrev,
    ITcVnContainer*         ipSrcKeyPoints,
    ITcVnImage*             ipSrcDescriptors,
    ITcVnContainer*         ipRefKeyPoints,
    ITcVnImage*             ipRefDescriptors,
    ULONG                   nRefImageWidth,
    ULONG                   nRefImageHeight,
    ITcVnContainer*&        ipEdgePoints,
    float                   fMaxDist,
    float                   fMaxKnnRatio,
    ETcVnNormalizationType  eNormType,
    TcVnMatrix3x3_LREAL&    aPerspectiveTransform,
    ULONG&                  nNumberOfGoodMatches,
```

```
    ULONG&                    nNumberOfInliers,
    ETcVnEstimationAlgorithm  eAlgorithm = EA_RANSAC,
    double                    fReprojThreshold = 3,
    ULONG                     nMaxIterations = 2000,
    double                    fConfidence = 0.995,
    ITcVnContainer**          pipGoodMatches = nullptr,
    ITcVnContainer**          pipInlierMask = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcKeyPoints	ITcVnContainer* [▶ 1890]		KeyPoints of the source image, e.g. (ContainerType_Vector_TcVnKeyPoint)
ipSrcDescriptors	ITcVnImage* [▶ 1925]		Descriptors of the source image KeyPoints
ipRefKeyPoints	ITcVnContainer* [▶ 1890]		KeyPoints of the reference image (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage* [▶ 1925]		Descriptors of the reference image KeyPoints
nRefImageWidth	ULONG		Width of the reference image in pixels
nRefImageHeight	ULONG		Height of the reference image in pixels
ipEdgePoints	ITcVnContainer* [▶ 1890]&		Returns a container with the 4 edge points of the reference image transformed into the coordinates of the source image (ContainerType_Vector_TcVnPoint2_REAL)
fMaxDist	float		Maximum allowed descriptor distance (-1 disables this filter criterion)
fMaxKnnRatio	float		Maximum allowed distance ratio between first and second best match ([0..1], -1 disables this filter criterion)
eNormType	ETcVnNormalizationType [▶ 1747]		Normalization type used for descriptor matching (HAMMING recommended for AKAZE, ORB, and BRISK. HAMMING2 is recommended for ORB if the ORB nBriefPoints parameter is 3 or 4.)
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 1713]&		Returns the perspective transformation matrix, which transforms the reference points to the source points
nNumberOfGoodMatches	ULONG&		Return the number of good matches
nNumberOfInliers	ULONG&		Return the number of inlier matches

Name	Type	Default	Description
eAlgorithm	ETcVnEstimationAlgorithm [▶ 1739]	EA_RANSAC	Estimation algorithm used for computing the perspective transformation between the two point sets
fReprojThreshold	double	3	Maximum allowed reprojection error to treat a point pair as an inlier (only used if eAlgorithm is RANSAC, RHO)
nMaxIterations	ULONG	2000	Maximum number of RANSAC iterations
fConfidence	double	0.995	Confidence (0..1)
pipGoodMatches	ITcVnContainer* [▶ 1890]*	nullptr	Returns a container with good matches (optional, set to 0 if not required; ContainerType_Vector_TcVnDMatch)
pipInlierMask	ITcVnContainer* [▶ 1890]*	nullptr	Returns a mask marking the inliers (optional, set to 0 if not required; ContainerType_Vector_SINT; only for RANSAC, LMEDS)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.18.3 FindReferenceKeyPointsInImageAKAZE

Searches a reference image, represented by its AKAZE keypoints and corresponding descriptors, in a given source image.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT FindReferenceKeyPointsInImageAKAZE(
    HRESULT                 hrPrev,
    ITcVnImage*             ipSrcImage,
    ITcVnContainer*         ipRefKeyPoints,
    ITcVnImage*             ipRefDescriptors,
    ULONG                   nRefImageWidth,
    ULONG                   nRefImageHeight,
    ITcVnContainer*&        ipEdgePoints,
    float                   fMaxDist,
    float                   fMaxKnnRatio,
    TcVnMatrix3x3_LREAL&    aPerspectiveTransform,
    ULONG&                  nNumberOfGoodMatches,
```

```
    ULONG&                    nNumberOfInliers,
    ETcVnNormalizationType    eNormType = NT_HAMMING,
    ITcVnImage*               ipSrcImageMask = nullptr,
    TcVnParamsAKAZE&          stAkazeParams = {FDT_AKAZE_MLDB, 0, 3, 0.001f, 2, 1, DT1_KAZE_PM_G2},
    ETcVnEstimationAlgorithm  eAlgorithm = EA_RANSAC,
    double                    fReprojThreshold = 3,
    ULONG                     nMaxIterations = 2000,
    double                    fConfidence = 0.995
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipRefKeyPoints	ITcVnContainer* [▶ 1890]		KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsAKAZE (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage* [▶ 1925]		Descriptors of the reference image KeyPoints, e.g. computed with F_VN_KeyPointsAndDescriptorsAKAZE
nRefImageWidth	ULONG		Width of the reference image in pixels
nRefImageHeight	ULONG		Height of the reference image in pixels
ipEdgePoints	ITcVnContainer* [▶ 1890]&		Returns a container with the 4 edge points of the reference image transformed into ipSrcImage coordinates (ContainerType_Vector_TcVnPoint2_REAL)
fMaxDist	float		Maximum allowed descriptor distance (-1 disables this filter criterion)
fMaxKnnRatio	float		Maximum allowed distance ratio between first and second best match ([0..1], -1 disables this filter criterion)
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 1713]&		Returns the perspective transformation matrix, which transforms the reference points to the source points
nNumberOfGoodMatches	ULONG&		Return the number of good matches
nNumberOfInliers	ULONG&		Return the number of inlier matches
eNormType	ETcVnNormalizationType [▶ 1747]	NT_HAMMING	Normalization type used for descriptor matching
ipSrcImageMask	ITcVnImage* [▶ 1925]	nullptr	Mask to specify, where to look for keypoints in ipSrcImage (set to 0 if no mask required)

Name	Type	Default	Description
stAkazeParams	TcVnParamsAKAZE [▶ 1771]&	{FDT_AKAZE_MLDB, 0, 3, 0.001f, 2, 1, DT1_KAZE_PM_G2}	Parameters to configure the keypoint and descriptor computation for ipSrcImage (resulting descriptors must be compatible to ipRefDescriptors!)
eAlgorithm	ETcVnEstimationAlgorithm [▶ 1739]	EA_RANSAC	Estimation algorithm used for computing the perspective transformation between the two point sets
fReprojThreshold	double	3	Maximum allowed reprojection error to treat a point pair as an inlier (only used if eAlgorithm is RANSAC, RHO)
nMaxIterations	ULONG	2000	Maximum number of RANSAC iterations
fConfidence	double	0.995	Confidence (0..1)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.18.4 FindReferenceKeyPointsInImageBRISK

Searches a reference image, represented by its BRISK keypoints and corresponding descriptors, in a given source image.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT FindReferenceKeyPointsInImageBRISK(
    HRESULT                   hrPrev,
    ITcVnImage*               ipSrcImage,
    ITcVnContainer*           ipRefKeyPoints,
    ITcVnImage*               ipRefDescriptors,
    ULONG                     nRefImageWidth,
    ULONG                     nRefImageHeight,
    ITcVnContainer*&          ipEdgePoints,
    float                     fMaxDist,
    float                     fMaxKnnRatio,
    TcVnMatrix3x3_LREAL&      aPerspectiveTransform,
    ULONG&                    nNumberOfGoodMatches,
    ULONG&                    nNumberOfInliers,
    ETcVnNormalizationType    eNormType = NT_HAMMING,
    ITcVnImage*               ipSrcImageMask = nullptr,
    TcVnParamsBRISK&          stBriskParams = {30, 3, 1},
    ETcVnEstimationAlgorithm  eAlgorithm = EA_RANSAC,
    double                    fReprojThreshold = 3,
```

```
    ULONG                   nMaxIterations = 2000,
    double                  fConfidence = 0.995
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipRefKeyPoints	ITcVnContainer* [▶ 1890]		KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsBRISK (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage* [▶ 1925]		Descriptors of the reference image KeyPoints, e.g. computed with F_VN_KeyPointsAndDescriptorsBRISK
nRefImageWidth	ULONG		Width of the reference image in pixels
nRefImageHeight	ULONG		Height of the reference image in pixels
ipEdgePoints	ITcVnContainer* [▶ 1890]&		Returns a container with the 4 edge points of the reference image transformed into ipSrcImage coordinates (ContainerType_Vector_TcVnPoint2_REAL)
fMaxDist	float		Maximum allowed descriptor distance (-1 disables this filter criterion)
fMaxKnnRatio	float		Maximum allowed distance ratio between first and second best match ([0..1], -1 disables this filter criterion)
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 1713]&		Returns the perspective transformation matrix, which transforms the reference points to the source points
nNumberOfGoodMatches	ULONG&		Return the number of good matches
nNumberOfInliers	ULONG&		Return the number of inlier matches
eNormType	ETcVnNormalizationType [▶ 1747]	NT_HAMMING	Normalization type used for descriptor matching
ipSrcImageMask	ITcVnImage* [▶ 1925]	nullptr	Mask to specify, where to look for keypoints in ipSrcImage (set to 0 if no mask required)

Name	Type	Default	Description
stBriskParams	TcVnParamsBRISK [▶ 1775]&	{30, 3, 1}	Parameters to configure the keypoint and descriptor computation for ipSrcImage (resulting descriptors must be compatible to ipRefDescriptors!)
eAlgorithm	ETcVnEstimationAlgorithm [▶ 1739]	EA_RANSAC	Estimation algorithm used for computing the perspective transformation between the two point sets
fReprojThreshold	double	3	Maximum allowed reprojection error to treat a point pair as an inlier (only used if eAlgorithm is RANSAC, RHO)
nMaxIterations	ULONG	2000	Maximum number of RANSAC iterations
fConfidence	double	0.995	Confidence (0..1)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.18.5 FindReferenceKeyPointsInImageORB

Searches a reference image, represented by its ORB keypoints and corresponding descriptors, in a given source image.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT FindReferenceKeyPointsInImageORB(
    HRESULT                  hrPrev,
    ITcVnImage*              ipSrcImage,
    ITcVnContainer*          ipRefKeyPoints,
    ITcVnImage*              ipRefDescriptors,
    ULONG                    nRefImageWidth,
    ULONG                    nRefImageHeight,
    ITcVnContainer*&         ipEdgePoints,
    float                    fMaxDist,
    float                    fMaxKnnRatio,
    TcVnMatrix3x3_LREAL&     aPerspectiveTransform,
    ULONG&                   nNumberOfGoodMatches,
    ULONG&                   nNumberOfInliers,
    ETcVnNormalizationType   eNormType = NT_HAMMING,
    ITcVnImage*              ipSrcImageMask = nullptr,
    TcVnParamsORB&           stOrbParams = {500, 1.2f, 8, 31, 0, 2, FST_ORB_HARRIS, 31, 20},
    ETcVnEstimationAlgorithm eAlgorithm = EA_RANSAC,
    double                   fReprojThreshold = 3,
```

```
    ULONG                   nMaxIterations = 2000,
    double                  fConfidence = 0.995
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipRefKeyPoints	ITcVnContainer* [▶ 1890]		KeyPoints of the reference image, e.g. computed with F_VN_KeyPointsAndDescriptorsORB (ContainerType_Vector_TcVnKeyPoint)
ipRefDescriptors	ITcVnImage* [▶ 1925]		Descriptors of the reference image KeyPoints, e.g. computed with F_VN_KeyPointsAndDescriptorsORB
nRefImageWidth	ULONG		Width of the reference image in pixels
nRefImageHeight	ULONG		Height of the reference image in pixels
ipEdgePoints	ITcVnContainer* [▶ 1890]&		Returns a container with the 4 edge points of the reference image transformed into ipSrcImage coordinates (ContainerType_Vector_TcVnPoint2_REAL)
fMaxDist	float		Maximum allowed descriptor distance (-1 disables this filter criterion)
fMaxKnnRatio	float		Maximum allowed distance ratio between first and second best match ([0..1], -1 disables this filter criterion)
aPerspectiveTransform	TcVnMatrix3x3_LREAL [▶ 1713]&		Returns the perspective transformation matrix, which transforms the reference points to the source points
nNumberOfGoodMatches	ULONG&		Return the number of good matches
nNumberOfInliers	ULONG&		Return the number of inlier matches
eNormType	ETcVnNormalizationType [▶ 1747]	NT_HAMMING	Normalization type used for descriptor matching (HAMMING recommended for ORB, HAMMING2 if the ORB nBriefPoints parameter is 3 or 4)

Name	Type	Default	Description
ipSrcImageMask	ITcVnImage* [▶ 1925]	nullptr	Mask to specify, where to look for keypoints in ipSrcImage (set to 0 if no mask required)
stOrbParams	TcVnParamsORB [▶ 1778]&	{500, 1.2f, 8, 31, 0, 2, FST_ORB_HARRIS, 31, 20}	Parameters to configure the keypoint and descriptor computation for ipSrcImage (resulting descriptors must be compatible to ipRefDescriptors!)
eAlgorithm	ETcVnEstimationAlgorithm [▶ 1739]	EA_RANSAC	Estimation algorithm used for computing the perspective transformation between the two point sets
fReprojThreshold	double	3	Maximum allowed reprojection error to treat a point pair as an inlier (only used if eAlgorithm is RANSAC, RHO)
nMaxIterations	ULONG	2000	Maximum number of RANSAC iterations
fConfidence	double	0.995	Confidence (0..1)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.18.6 GetMatchCoordinates

Return the coordinates of keypoints that match each other.

Syntax

Definition:

```
HRESULT GetMatchCoordinates(
    HRESULT          hrPrev,
    ITcVnContainer*  ipQueryKeyPoints,
    ITcVnContainer*  ipTrainKeyPoints,
    ITcVnContainer*  ipMatches,
    ITcVnContainer*& ipQueryCoordinates,
    ITcVnContainer*& ipTrainCoordinates
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipQueryKeyPoints	ITcVnContainer* [▶ 1890]	Container with query Keypoints (i.e. keypoints of a reference/ template image) of type CTcVnContainer_Vector_TcVnKeyPoint
ipTrainKeyPoints	ITcVnContainer* [▶ 1890]	Container with training Keypoints (i.e. keypoints of a source/input image) of type CTcVnContainer_Vector_TcVnKeyPoint
ipMatches	ITcVnContainer* [▶ 1890]	Container with matches between query and train Keypoints (CTcVnContainer_Vector_TcVnDMatch).
ipQueryCoordinates	ITcVnContainer* [▶ 1890]&	Returns a container with the coordinates of keypoints from the query Keypoints that exist in ipMatches (CTcVnContainer_Vector_TcVnPoint2_REAL)
ipTrainCoordinates	ITcVnContainer* [▶ 1890]&	Returns a container with the coordinates of keypoints from the training Keypoints that exist in ipMatches (CTcVnContainer_Vector_TcVnPoint2_REAL)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.18.7 KeyPointsAGAST

Detects keypoints using the AGAST method.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT KeyPointsAGAST(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage,
    ITcVnContainer*& ipKeyPoints,
```

```
    ITcVnImage*      ipMask = nullptr,
    TcVnParamsAGAST& stParams = {10, true, KDT_AGAST_9_16}
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipKeyPoints	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipMask	ITcVnImage* [▶ 1925]	nullptr	Mask to specify, where to look for keypoints (set to 0 if no mask required)
stParams	TcVnParamsAGAST [▶ 1770]&	{10, true, KDT_AGAST_9_16}	Additional expert parameters

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.18.8 KeyPointsAndDescriptorsAKAZE

Detects keypoints and compute descriptors using the AKAZE method.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT KeyPointsAndDescriptorsAKAZE(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage,
    ITcVnContainer*& ipKeyPoints,
    ITcVnImage*&     ipDescriptors,
    ITcVnImage*      ipMask = nullptr,
    TcVnParamsAKAZE& stParams = {FDT_AKAZE_MLDB, 0, 3, 0.001f, 2, 1, DT1_KAZE_PM_G2}
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipKeyPoints	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	ITcVnImage* [▶ 1925]&		Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
ipMask	ITcVnImage* [▶ 1925]	nullptr	Mask to specify, where to look for keypoints (set to 0 if no mask required)
stParams	TcVnParamsAKAZE [▶ 1771]&	{FDT_AKAZE_MLDB, 0, 3, 0.001f, 2, 1, DT1_KAZE_PM_G2}	Additional expert parameters

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.18.9 KeyPointsAndDescriptorsBRISK

Detects keypoints and compute descriptors using the BRISK method.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT KeyPointsAndDescriptorsBRISK(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage,
    ITcVnContainer*& ipKeyPoints,
    ITcVnImage*&     ipDescriptors,
    ITcVnImage*      ipMask = nullptr,
    TcVnParamsBRISK& stParams = {30, 3, 1}
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipKeyPoints	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	ITcVnImage* [▶ 1925]&		Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
ipMask	ITcVnImage* [▶ 1925]	nullptr	Mask to specify, where to look for keypoints (set to 0 if no mask required)
stParams	TcVnParamsBRISK [▶ 1775]&	{30, 3, 1}	Additional expert parameters

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.18.10 KeyPointsAndDescriptorsKAZE

Detects keypoints and compute descriptors using the KAZE method.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT KeyPointsAndDescriptorsKAZE(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage,
    ITcVnContainer*& ipKeyPoints,
    ITcVnImage*&     ipDescriptors,
    ITcVnImage*      ipMask = nullptr,
    TcVnParamsKAZE&  stParams = {false, false, 0.001f, 4, 2, DT1_KAZE_PM_G2}
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipKeyPoints	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	ITcVnImage* [▶ 1925]&		Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
ipMask	ITcVnImage* [▶ 1925]	nullptr	Mask to specify, where to look for keypoints (set to 0 if no mask required)
stParams	TcVnParamsKAZE [▶ 1777]&	{false, false, 0.001f, 4, 2, DT1_KAZE_PM_G2}	Additional expert parameters

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.18.11 KeyPointsAndDescriptorsORB

Detect keypoints and compute descriptors using the ORB method.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT KeyPointsAndDescriptorsORB(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage,
    ITcVnContainer*& ipKeyPoints,
    ITcVnImage*&     ipDescriptors,
    ITcVnImage*      ipMask = nullptr,
    TcVnParamsORB&   stParams = {500, 1.2f, 8, 31, 0, 2, FST_ORB_HARRIS, 31, 20}
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipKeyPoints	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipDescriptors	ITcVnImage* [▶ 1925]&		Descriptor image (set to 0 if not required; 1 descriptor in each row; An appropriate image will be created if required.)
ipMask	ITcVnImage* [▶ 1925]	nullptr	Mask to specify, where to look for keypoints (set to 0 if no mask required)
stParams	TcVnParamsORB [▶ 1778]&	{500, 1.2f, 8, 31, 0, 2, FST_ORB_HARRIS, 31, 20}	Additional expert parameters

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.18.12 KeyPointsFAST

Detects keypoints using the FAST method.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT KeyPointsFAST(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage,
    ITcVnContainer*& ipKeyPoints,
    ITcVnImage*      ipMask = nullptr,
    TcVnParamsFAST&  stParams = {10, true, KDT_FAST_9_16}
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipKeyPoints	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipMask	ITcVnImage* [▶ 1925]	nullptr	Mask to specify, where to look for keypoints (set to 0 if no mask required)
stParams	TcVnParamsFAST [▶ 1776]&	{10, true, KDT_FAST_9_16}	Additional expert parameters

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.18.13 KeyPointsGFTT

Detects keypoints using the GFTT method, which detects strong corners.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT KeyPointsGFTT(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage,
    ITcVnContainer*& ipKeyPoints,
    ITcVnImage*      ipMask = nullptr,
    TcVnParamsGFTT&  stParams = {1000, 0.01, 1, 3, false, 0.04}
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipKeyPoints	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipMask	ITcVnImage* [▶ 1925]	nullptr	Mask to specify, where to look for keypoints (set to 0 if no mask required)
stParams	TcVnParamsGFTT [▶ 1776]&	{1000, 0.01, 1, 3, false, 0.04}	Additional expert parameters

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.18.14 KeyPointsMSER

Detects keypoints using the MSER method.

Syntax

Definition:

```
HRESULT KeyPointsMSER(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage,
    ITcVnContainer*& ipKeyPoints,
    ITcVnImage*      ipMask = nullptr,
    TcVnParamsMSER&  stParams = {5, 60, 14400, 0.25, 0.2, 200, 1.01, 0.003, 5}
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipKeyPoints	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipMask	ITcVnImage* [▶ 1925]	nullptr	Mask to specify, where to look for keypoints (set to 0 if no mask required)
stParams	TcVnParamsMSER [▶ 1777]&	{5, 60, 14400, 0.25, 0.2, 200, 1.01, 0.003, 5}	Additional expert parameters

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.18.15 KeyPointsSB

Detects keypoints using a Simple Blob method - several iterations apply different thresholds to source image - connected components (blobs) are detected - the center and radius of the blobs are returned as keypoints Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT KeyPointsSB(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage,
    ITcVnContainer*& ipKeyPoints,
    ITcVnImage*      ipMask = nullptr,
    TcVnParamsSB&    stParams = {true, false, false, false, false, 25, 15000, 0, 1, 255, 0, 1, 0, 1,
5, 2, 30, 225, 10}
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipKeyPoints	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the keypoints (ContainerType_Vector_TcVnKeyPoint; Non-zero interface pointers are reused.)
ipMask	ITcVnImage* [▶ 1925]	nullptr	Mask to specify, where to look for keypoints (optional, set to 0 if not required)
stParams	TcVnParamsSB [▶ 1779]&	{true, false, false, false, false, 25, 15000, 0, 1, 255, 0, 1, 0, 1, 5, 2, 30, 225, 10}	Several parameters to filter the detected blobs

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.18.16 MatchDescriptorsBF

Match descriptors using a brute force approach.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT MatchDescriptorsBF(
    HRESULT                hrPrev,
    ITcVnImage*            ipQueryDescriptors,
    ITcVnImage*            ipTrainDescriptors,
    ITcVnContainer*&       ipMatches,
    ITcVnImage*            ipMask = nullptr,
    ETcVnNormalizationType eNormType = NT_L2,
    bool                   bCrossCheck = false
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipQueryDescriptors	ITcVnImage* [▶ 1925]		Query descriptors (i.e. descriptors of a reference/ template image)
ipTrainDescriptors	ITcVnImage* [▶ 1925]		Training descriptors (i.e. descriptors of a source/ input image)
ipMatches	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the descriptor matches (ContainerType_Vector_TcVnDMatch; Non-zero interface pointers are reused.)
ipMask	ITcVnImage* [▶ 1925]	nullptr	Mask to specify permissable matches, i.e. query[i] can be matched with train[j] only if mask[i][j] != 0 (set to 0 if no mask required)
eNormType	ETcVnNormalizationType [▶ 1747]	NT_L2	Normalization type (only L1, L2, L2SQR, HAMMING, HAMMING2 supported). HAMMING2 should be used for ORB descriptors if the ORB nBriefPoints parameter is 3 or 4)
bCrossCheck	bool	false	If true, only consistent matches are returned, i.e. query->train and train->query detect the same match

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.18.17 MatchDescriptorsFlannLsh

Match descriptors using a FLANN based approach with LSH index.

Syntax

Definition:

```
HRESULT MatchDescriptorsFlannLsh(
    HRESULT          hrPrev,
    ITcVnImage*      ipQueryDescriptors,
    ITcVnImage*      ipTrainDescriptors,
    ITcVnContainer*& ipMatches,
    ULONG            nTableNumber = 12,
    ULONG            nKeySize = 20,
    ULONG            nMultiProbeLevel = 2,
    ULONG            nChecks = 32
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipQueryDescriptors	ITcVnImage* [▶ 1925]		Query descriptors (i.e. descriptors of a reference/ template image)
ipTrainDescriptors	ITcVnImage* [▶ 1925]		Training descriptors (i.e. descriptors of a source/ input image)
ipMatches	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the descriptor matches (ContainerType_Vector_TcVnDMatch; Non-zero interface pointers are reused.)
nTableNumber	ULONG	12	Number of tables
nKeySize	ULONG	20	Key size
nMultiProbeLevel	ULONG	2	Multi-probe level
nChecks	ULONG	32	Maximum number of visited leafs when searching for neighbors

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.18.18 MatchDescriptorsKnnBF

Match descriptors (k nearest neighbors) using a brute force approach.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT MatchDescriptorsKnnBF(
    HRESULT                 hrPrev,
    ITcVnImage*             ipQueryDescriptors,
    ITcVnImage*             ipTrainDescriptors,
    ITcVnContainer*&        ipMatches,
    ULONG                   nK,
    ITcVnImage*             ipMask = nullptr,
    bool                    bCompactResult = false,
    ETcVnNormalizationType  eNormType = NT_L2,
    bool                    bCrossCheck = false
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipQueryDescriptors	ITcVnImage* [▶ 1925]		Query descriptors (i.e. descriptors of a reference/ template image)
ipTrainDescriptors	ITcVnImage* [▶ 1925]		Training descriptors (i.e. descriptors of a source/ input image)
ipMatches	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the descriptor matches (ContainerType_Vector_Vector_TcVnDMatch; Non-zero interface pointers are reused.)
nK	ULONG		Number of required best train matches for each query descriptor (i.e. nk := 2)
ipMask	ITcVnImage* [▶ 1925]	nullptr	Mask to specify permissable matches, i.e. query[i] can be matched with train[j] only if mask[i][j] != 0 (set to 0 if no mask required)
bCompactResult	bool	false	If true, matches vector does not contain entries for fully masked-out query descriptors
eNormType	ETcVnNormalizationType [▶ 1747]	NT_L2	Normalization type (only L1, L2, L2SQR, HAMMING, HAMMING2 supported). HAMMING2 should be used for ORB descriptors if the ORB nBriefPoints parameter is 3 or 4)
bCrossCheck	bool	false	If true, only consistent matches are returned, i.e. query->train and train->query detect the same match (only used if nK = 1)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.18.19 MatchDescriptorsKnnFlannLsh

Match descriptors (k nearest neighbors) using a FLANN based approach with LSH index.

Syntax

Definition:

```
HRESULT MatchDescriptorsKnnFlannLsh(
    HRESULT          hrPrev,
    ITcVnImage*      ipQueryDescriptors,
    ITcVnImage*      ipTrainDescriptors,
    ITcVnContainer*& ipMatches,
    ULONG            nK,
    ULONG            nTableNumber = 12,
    ULONG            nKeySize = 20,
    ULONG            nMultiProbeLevel = 2,
    ULONG            nChecks = 32
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipQueryDescriptors	ITcVnImage* [▶ 1925]		Query descriptors (i.e. descriptors of a reference/ template image)
ipTrainDescriptors	ITcVnImage* [▶ 1925]		Training descriptors (i.e. descriptors of a source/ input image)
ipMatches	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the descriptor matches (ContainerType_Vector_Vector_TcVnDMatch; Non-zero interface pointers are reused.)
nK	ULONG		Number of required best train matches for each query descriptor (i.e. nk := 2)
nTableNumber	ULONG	12	Number of tables
nKeySize	ULONG	20	Key size
nMultiProbeLevel	ULONG	2	Multi-probe level
nChecks	ULONG	32	Maximum number of visited leafs when searching for neighbors

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.18.20 RegionsMSER

Detects regions using the MSER method.

Syntax

Definition:

```
HRESULT RegionsMSER(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage,
    ITcVnContainer*& ipRegions,
    ITcVnContainer*& ipBoundingBoxes,
    TcVnParamsMSER&  stParams = {5, 60, 14400, 0.25, 0.2, 200, 1.01, 0.003, 5}
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image
ipRegions	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the region points (ContainerType_Vector_Vector_TcVnPoint2_DINT; Non-zero interface pointers are reused.)
ipBoundingBoxes	ITcVnContainer* [▶ 1890]&		Returns a container which is filled with the region bounding boxes (ContainerType_Vector_TcVnRectangle_DINT; Non-zero interface pointers are reused.)
stParams	TcVnParamsMSER [▶ 1777]&	{5, 60, 14400, 0.25, 0.2, 200, 1.01, 0.003, 5}	Additional expert parameters

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19 Machine Learning

Diese Gruppe enthält Funktionen zum Erstellen, Trainieren und Anwenden von Machine Learning Modellen sowie Funktionen zur Merkmalsvorverarbeitung. Eine kurze Übersicht der Anwendungsmöglichkeiten von Machine Learning [▶ 153] und weitere Beschreibungen finden Sie unter dem Softwarekonzept [▶ 146].

Funktionen zum Erstellen eines Modells:

- CreateBoostClassifier [▶ 2610]
- CreateKmppModel [▶ 2611]
- CreateKnnModel [▶ 2612]
- CreateLbgModel [▶ 2613]
- CreateLdaTransform [▶ 2615]
- CreateLdaTransformViaComponentNum [▶ 2616]
- CreateNbcModel [▶ 2617]
- CreatePcaTransform [▶ 2617]
- CreatePcaTransformViaComponentNum [▶ 2618]
- CreatePcaTransformViaVariance [▶ 2619]
- CreateRTreesModel [▶ 2620]
- CreateStaModel [▶ 2622]
- CreateSvmModel [▶ 2625]
- CreateSvmSgdClassifier [▶ 2628]

Funktionen zum Trainieren eines Modells:

- TrainBatch [▶ 2644]
- TrainBatchClusters [▶ 2645]
- TrainSample [▶ 2646]
- TrainSampleClass [▶ 2647]
- TrainSampleCluster [▶ 2648]
- TrainSampleScalar [▶ 2648]
- TrainSampleVector [▶ 2649]

Funktionen zum Erhalt von Informationen zu den Cluster Modellen:

- GetClusterCenter [▶ 2633]
- GetClusterNum [▶ 2634]

Funktionen zum Anwenden von Modellen auf einem Sample:

- GetSampleCluster [▶ 2636]
- GetSampleNovelty [▶ 2636]
- PredictSampleClass [▶ 2641]

- PredictSampleScalar [▶ 2642]
- PredictSampleVector [▶ 2643]

Funktionen zum Anwenden von Modellen auf einem Batch:

- GetBatchClusters [▶ 2632]
- GetBatchNovelties [▶ 2632]
- PredictBatch [▶ 2640]

Funktionen zum Anwenden von Modellen zur Merkmalstransformation / Dimensionsreduktion:

- FeatureTransform [▶ 2631]
- InverseFeatureTransform [▶ 2639]

Funktionen zur Merkmalsnormalisierung:

- GetFeatureScales [▶ 2635]
- FeatureScaling [▶ 2630]
- InverseFeatureScaling [▶ 2637]
- InverseFeatureScaling_REAL [▶ 2639]

6.2.3.19.1 CreateBoostClassifier

Create a Boost classifier. The initial reference count is set to one if a new model is created and kept, otherwise. The Boost classifier is only applicable to binary classification problems. It learns to distinguish between samples labelled with two user-defined class labels by incrementally adding weak classifiers to improve the classification results. Models of this type neither support on-line training (sample by sample) nor retraining.

Syntax

Definition:

```
HRESULT CreateBoostClassifier(
    HRESULT                 hrPrev,
    ITcVnMlModel*&          ipMlModel,
    ETcVnBoostClassifierType eType = BCT_REAL,
    ULONG                   nMaxDepth = 1,
    ULONG                   nMinSamples = 10,
    ULONG                   nWeakClassifiers = 100,
    double                  fWeightTrimRate = 0.95,
    ITcVnContainer*         ipClassPriors = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1933]&		Returns the created model (Non-zero interface pointers are reused.)
eType	ETcVnBoostClassifierType [▶ 1716]	BCT_REAL	Learning algorithm type (default: TCVN_BCT_REAL)
nMaxDepth	ULONG	1	Maximum tree depth (default: 1)
nMinSamples	ULONG	10	Minimum number of samples within a node required for splitting (default: 10)
nWeakClassifiers	ULONG	100	Number of weak classifiers (default: 100)
fWeightTrimRate	double	0.95	Weight threshold used during training (off: 0; default: 0.95).
ipClassPriors	ITcVnContainer* [▶ 1890]	nullptr	Class priors (ContainerType_Vector_REAL or ContainerType_Vector_LREAL; only for classifiers; optional, set to 0 if not required or not allowed; default: 0)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.2 CreateKmppModel

Create a k-means++ model of the specified type. The initial reference count is set to one if a new model is created and kept, otherwise. Models of this type neither support on-line training (sample by sample) nor retraining.

Syntax

Definition:

```
HRESULT CreateKmppModel(
    HRESULT                 hrPrev,
    ITcVnMlModel*&          ipMlModel,
    ETcVnPrototypeClusterer eKmppType,
    ULONG                   nK,
    bool                    bDoublePrecision = false,
    ULONG                   nMaxIterations = 0,
    double                  fEpsilon = 0.0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1933]&		Returns the created model (Non-zero interface pointers are reused.)
eKmppType	ETcVnPrototypeClusterer [▶ 1752]		k-means++ model type
nK	ULONG		Parameter k (number of clusters)
bDoublePrecision	bool	false	If true, the model is generated with double precision (LREAL) instead of single precision (REAL). (default: FALSE)
nMaxIterations	ULONG	0	Maximum number of iterations (triggers the usage of the default value of 10 if it equals 0)
fEpsilon	double	0.0	Maximum allowed difference of the error between two successive iterations (triggers the usage of the default value of 0.001 if it equals 0)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.3 CreateKnnModel

Create a k-nearest neighbors model of the specified type. The initial reference count is set to one if a new model is created and kept, otherwise. Models of this type support on-line training (sample by sample) and retraining. Predictions can only be scalar.

Syntax

Definition:

```
HRESULT CreateKnnModel(
    HRESULT         hrPrev,
    ITcVnMlModel*& ipMlModel,
    ETcVnKnn        eKnnType,
    ULONG           nK
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1933]&	Returns the created model (Non-zero interface pointers are reused.)
eKnnType	ETcVnKnn [▶ 1745]	k-nearest neighbors model type
nK	ULONG	Parameter k used for prediction (number of considered neighbors)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.4 CreateLbgModel

Create a LBG model of the specified type. The initial reference count is set to one if a new model is created and kept, otherwise. Models of this type neither support on-line training (sample by sample) nor retraining.

Syntax

Definition:

```
HRESULT CreateLbgModel(
    HRESULT                  hrPrev,
    ITcVnMlModel*&           ipMlModel,
    ETcVnPrototypeClusterer eLbgType,
    ULONG                    nMaxClusters,
    double                   fMaxClusterRadius,
    bool                     bSingleSplitSteps,
    bool                     bDoublePrecision = false,
    ULONG                    nMaxIterations = 0,
    double                   fEpsilon = 0.0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1933]&		Returns the created model (Non-zero interface pointers are reused.)
eLbgType	ETcVnPrototypeClusterer [▶ 1752]		LBG model type
nMaxClusters	ULONG		Maximum number of clusters
fMaxClusterRadius	double		Maximum allowed radius (L2 norm) of a single cluster, i.e. clusters with a higher radius will be split into smaller ones, until a global number of nMaxClusters is reached.
bSingleSplitSteps	bool		If true, the global optimization is always run after a single cluster has been split. If false, several clusters are split within the same step before applying the global optimization. Applying the global optimization less often is faster, but can lead to less optimal results, especially having 2 nearby clusters that could be represented by 1.
bDoublePrecision	bool	false	If true, the model is generated with double precision (LREAL) instead of single precision (REAL). (default: FALSE)
nMaxIterations	ULONG	0	Maximum number of iterations (triggers the usage of the default value of 10 if it equals 0)
fEpsilon	double	0.0	Maximum allowed difference of the error between two successive iterations (triggers the usage of the default value of 0.001 if it equals 0)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.5 CreateLdaTransform

Create an LDA-based feature transform from the provided data. The number of samples must be >= the number of features and the number of classes must be >= 2. The initial reference count is set to one if a new model is created and kept, otherwise.

Syntax

Definition:

```
HRESULT CreateLdaTransform(
    HRESULT         hrPrev,
    ITcVnMlModel*&  ipMlModel,
    ITcUnknown*     ipSamples,
    ITcVnContainer* ipClasses
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1933]&	Returns the created feature transform (Non-zero interface pointers are reused.)
ipSamples	ITcUnknown* [▶ 1939]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipClasses	ITcVnContainer* [▶ 1890]	Class labels corresponding to the input samples (ContainerType_Vector_DINT)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.6 CreateLdaTransformViaComponentNum

Create an LDA-based feature transform from the provided data where the number of components to be used is predefined. The number of samples must be >= the number of features and the number of classes must be >= 2. The initial reference count is set to one if a new model is created and kept, otherwise.

Syntax

Definition:

```
HRESULT CreateLdaTransformViaComponentNum(
    HRESULT          hrPrev,
    ITcVnMlModel*&   ipMlModel,
    ITcUnknown*      ipSamples,
    ITcVnContainer*  ipClasses,
    ULONG            nComponentNum
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1933]&	Returns the created feature transform (Non-zero interface pointers are reused.)
ipSamples	ITcUnknown* [▶ 1939]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipClasses	ITcVnContainer* [▶ 1890]	Class labels corresponding to the input samples (ContainerType_Vector_DINT)
nComponentNum	ULONG	Number of components (nComponentNum must be smaller than the number of classes. A value of 0 results in the automatic computation of the maximum number of components.)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.7 CreateNbcModel

Create a normal Bayes classifier of the specified type. The initial reference count is set to one if a new model is created and kept, otherwise. In order to train normal Bayes classifiers, a sufficiently high number of samples is required for each class. It depends on the number of features and the distribution of the data. Hence, it needs to be tested for each application. Models of this type do not support on-line training (sample by sample). For the retraining of such classifier models, the set of presented classes must be identical to the previous learning steps. Otherwise, an exception is raised.

Syntax

Definition:

```
HRESULT CreateNbcModel(
    HRESULT        hrPrev,
    ITcVnMlModel*& ipMlModel,
    ETcVnNbc       eNbcType
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1933]&	Returns the created model (Non-zero interface pointers are reused.)
eNbcType	ETcVnNbc [▶ 1747]	Normal Bayes classifier type

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.8 CreatePcaTransform

Create a PCA-based feature transform from the provided data. The maximum number of principal components that can be computed equals the minimum of the number of samples and the number of features. The initial reference count is set to one if a new model is created and kept, otherwise.

Syntax

Definition:

```
HRESULT CreatePcaTransform(
    HRESULT        hrPrev,
    ITcVnMlModel*& ipMlModel,
    ITcUnknown*    ipSamples
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1933]&	Returns the created feature transform (Non-zero interface pointers are reused.)
ipSamples	ITcUnknown* [▶ 1939]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.9 CreatePcaTransformViaComponentNum

Create a PCA-based feature transform from the provided data where the number of prinicpal components to be used is predefined. The maximum number of principal components that can be computed equals the minimum of the number of samples and the number of features. The initial reference count is set to one if a new model is created and kept, otherwise.

Syntax

Definition:

```
HRESULT CreatePcaTransformViaComponentNum(
    HRESULT        hrPrev,
    ITcVnMlModel*& ipMlModel,
    ITcUnknown*    ipSamples,
    ULONG          nComponentNum
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1933]&	Returns the created feature transform (Non-zero interface pointers are reused.)
ipSamples	ITcUnknown* [▶ 1939]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
nComponentNum	ULONG	Number of principal components (A value of 0 results in the automatic computation of the maximum number of components.)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.10 CreatePcaTransformViaVariance

Create a PCA-based feature transform from the provided data based on a given fraction of variance to be retained. The maximum number of principal components that can be computed equals the minimum of the number of samples and the number of features. The initial reference count is set to one if a new model is created and kept, otherwise.

Syntax

Definition:

```
HRESULT CreatePcaTransformViaVariance(
    HRESULT        hrPrev,
    ITcVnMlModel*& ipMlModel,
    ITcUnknown*    ipSamples,
    double         fRetainedVariance
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1933]&	Returns the created feature transform (Non-zero interface pointers are reused.)
ipSamples	ITcUnknown* [▶ 1939]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
fRetainedVariance	double	Fraction of variance that is to be retained by the PCA (A value of 1.0 signifies 100%.)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.11 CreateRTreesModel

Create a random trees model of the specified type. The initial reference count is set to one if a new model is created and kept, otherwise. Models of this type neither support on-line training (sample by sample) nor retraining. Predictions can only be scalar.

Syntax

Definition:

```
HRESULT CreateRTreesModel(
    HRESULT         hrPrev,
    ITcVnMlModel*&  ipMlModel,
    ETcVnRTrees     eRTreesType,
    ULONG           nMaxDepth = 5,
    ULONG           nMinSamples = 10,
    ULONG           nActiveVariables = 0,
    ULONG           nMaxIterations = 0,
    double          fEpsilon = 0.0,
    float           fRegressionAccuracy = 0,
    ITcVnContainer* ipClassPriors = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1933]&		Returns the created model (Non-zero interface pointers are reused.)
eRTreesType	ETcVnRTrees [▶ 1753]		Random trees model type
nMaxDepth	ULONG	5	Maximum tree depth (default: 5)
nMinSamples	ULONG	10	Minimum number of samples within a node required for splitting (default: 10)
nActiveVariables	ULONG	0	Number of variables considered for splitting (0 means sqrt(total number of variables); default: 0)
nMaxIterations	ULONG	0	Maximum number of iterations (disabled if it equals 0 and fEpsilon is different from 0.0; triggers the usage of the default value of 50 if nMaxIterations and fEpsilon equal 0)
fEpsilon	double	0.0	Maximum allowed difference of the error between two successive iterations (disabled if it equals 0.0 and nMaxIterations is different from 0; triggers the usage of the default value of 0.1 if nMaxIterations and fEpsilon equal 0)
fRegressionAccuracy	float	0	Termination criterion for regressors (only for regressors; set to default if not allowed; default: 0.0)
ipClassPriors	ITcVnContainer* [▶ 1890]	nullptr	Class priors (ContainerType_Vector_REAL or ContainerType_Vector_LREAL; only for classifiers; optional, set to 0 if not required or not allowed; default: 0)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.12 CreateStaModel

Create a Simplified TopoART neural network of the specified type. The initial reference count is set to one if a new model is created and kept, otherwise. Models of this type support on-line training (sample by sample), retraining, as well as scalar and vectorial predictions. It requires all input except class labels (i.e., samples and training outputs/predictions) to lie in the interval [0.0, 1.0]. The predictions of regressors need to be rescaled from the interval [0.0, 1.0] to their respective value range before usage. Depending on the parameter settings and the number of available training samples, repeated training with the same data may improve the results. Like other neural networks based on the Adaptive Resonance Theory (ART), Simplified TopoART neural networks are not prone to catastrophic inference and patricularly well-suited to incremental learning tasks. (additional expert function providing parameters for fine-tuning and noise reduction)

Syntax

Definition:

```
HRESULT CreateStaModel(
    HRESULT        hrPrev,
    ITcVnMlModel*& ipMlModel,
    ETcVnSta       eStaType,
    double         fRho,
    ULONG          nNu = 3,
    bool           bDoublePrecision = false,
    double         fBetaSbm = 0.0,
    ULONG          nPhi = 1,
    ULONG          nTau = 100
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1933]&		Returns the created model (Non-zero interface pointers are reused.)
eStaType	ETcVnSta [▶ 1754]		Simpified TopoART model type
fRho	double		Vigilance parameter (controls the number of neurons that are inserted and the maximum size of the formed categories; valid range: [0.0, 1.0]; suggested range: [0.8, 1.0))
nNu	ULONG	3	Number of neurons used for classification and prediction (must be larger than or equal to 1; suggested range: [1, 10); default: 3)
bDoublePrecision	bool	false	If true, the model is generated with double precision (LREAL) instead of single precision (REAL). (default: FALSE)
fBetaSbm	double	0.0	Learning rate of the second best-matching neuron (learning the second best-matching neuron keeps related categories closer together and might improve the results but may require a higher number of neurons and training runs; a value of 0.0 disables learning of the second best-matching neuron; valid range: [0.0, 1.0]; suggested range: [0.0, 0.5]); default: 0.0
nPhi	ULONG	1	Number of samples a neuron must have learnt to become permanent (required for noise reduction; works in conjunction with nTau; must be larger than or equal to 1; higher values intensify noise reduction; a value of 1 disables removal of neuron candidates; default: 1)

Name	Type	Default	Description
nTau	ULONG	100	Number of learning steps after which node removal is performed (required for noise reduction; works in conjunction with nPhi; must be large enough to allow neuron candidates representing non-noise samples to become the best-matching neuron at least nPhi times before node removal; default: 100)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.13 CreateSvmModel

Create an SVM model of the specified type. The initial reference count is set to one if a new model is created and kept, otherwise. Models of this type neither support on-line training (sample by sample) nor retraining. Predictions can only be scalar. (additional expert function for C support vector classifiers)

Syntax

Definition:

```
HRESULT CreateSvmModel(
    HRESULT            hrPrev,
    ITcVnMlModel*&     ipMlModel,
    ETcVnSvm           eSvmType,
    double             fC,
    double             fNu,
    double             fP,
    ETcVnSvmKernelType eKernelType,
    double             fGamma,
    double             fCoef0,
    double             fDegree,
    ULONG              nMaxIterations = 0,
    double             fEpsilon = 0.0,
    ITcVnContainer*    ipClassWeights = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1933]&		Returns the created model (Non-zero interface pointers are reused.)
eSvmType	ETcVnSvm [▶ 1755]		SVM model type
fC	double		Parameter C (required for TCVN_SVM_C_CLASSIFIER, TCVN_SVM_EPS_REGRESSOR, and TCVN_SVM_NU_REGRESSOR; ignored otherwise)
fNu	double		Parameter nu (required for TCVN_SVM_NU_CLASSIFIER, TCVN_SVM_NOVELTY_DETECTOR, and TCVN_SVM_NU_REGRESSOR; ignored otherwise)
fP	double		Parameter p (required for TCVN_SVM_EPS_REGRESSOR; ignored otherwise)
eKernelType	ETcVnSvmKernelType [▶ 1755]		Kernel type
fGamma	double		Parameter gamma (used by polynomial, RBF, sigmoid, and chi-squared kernels; ignored otherwise)
fCoef0	double		Parameter coef0 (used by polynomial and sigmoid kernels; ignored otherwise)
fDegree	double		Degree (used by polynomial kernels; ignored otherwise)
nMaxIterations	ULONG	0	Maximum number of iterations (disabled if it equals 0 and fEpsilon is different from 0.0; triggers the usage of the default value of 100000 if nMaxIterations and fEpsilon equal 0)

Name	Type	Default	Description
fEpsilon	double	0.0	Maximum allowed difference of the error between two successive iterations (disabled if it equals 0.0 and nMaxIterations is different from 0; triggers the usage of the default value of 0.00001 if nMaxIterations and fEpsilon equal 0)
ipClassWeights	ITcVnContainer* [▶ 1890]	nullptr	Class weights (ContainerType_Vector_REAL or ContainerType_Vector_LREAL; only valid if eSvmType equals TCVN_SVM_C_CLASSIFIER; optional, set to 0 if not required or not allowed; default: 0)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.14 CreateSvmSgdClassifier

Create a linear SVM classifier using stochastic gradient descent for training. The initial reference count is set to one if a new model is created and kept, otherwise. This SVM classifier is only applicable to binary classification problems. It learns a separating hyperplane between a class with label -1 and a class with label 1. These class labels are predefined. For training, any positive class labels are mapped to 1 and any negative class labels are mapped to -1. Models of this type neither support on-line training (sample by sample) nor retraining.

Syntax

Definition:

```
HRESULT CreateSvmSgdClassifier(
    HRESULT                          hrPrev,
    ITcVnMlModel*&                   ipMlModel,
    ETcVnSvmSgdClassifierType        eType = SSCT_ASGD,
    ETcVnSvmSgdClassifierMarginType  eMarginType = SSCMT_SOFT_MARGIN,
    float                            fMarginRegularization = 0.00001f,
    float                            fInitialStepSize = 0.05f,
    float                            fStepDecreasingPower = 0.75f,
    ULONG                            nMaxIterations = 0,
    double                           fEpsilon = 0.0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1933]&		Returns the created model (Non-zero interface pointers are reused.)
eType	ETcVnSvmSgdClassifierType [▶ 1756]	SSCT_ASGD	Learning algorithm type (default: TCVN_SSCT_ASGD)
eMarginType	ETcVnSvmSgdClassifierMarginType [▶ 1756]	SSCMT_SOFT_MARGIN	Margin type (default: TCVN_SSCMT_SOFT_MARGIN)
fMarginRegularization	float	0.00001f	Margin regularization parameter (default: 0.00001)
fInitialStepSize	float	0.05f	Initial step size (default: 0.05)
fStepDecreasingPower	float	0.75f	Power parameter (default: 0.75)
nMaxIterations	ULONG	0	Maximum number of iterations (disabled if it equals 0 and fEpsilon is different from 0.0; triggers the usage of the default value of 100000 if nMaxIterations and fEpsilon equal 0)
fEpsilon	double	0.0	Maximum allowed difference of the error between two successive iterations (disabled if it equals 0.0 and nMaxIterations is different from 0; triggers the usage of the default value of 0.00001 if nMaxIterations and fEpsilon equal 0)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.15 FeatureScaling

Apply a feature scaling to one or more sample(s) and get output range between [fA, fB] in the case of TCVN_FST1_MINMAX.

Syntax

Definition:

```
HRESULT FeatureScaling(
    HRESULT          hrPrev,
    ITcVnContainer*  ipSamples,
    ITcVnContainer*  ipScales,
    ITcVnContainer*& ipScaledSamples,
    double           fA = 0.0,
    double           fB = 1.0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSamples	ITcVnContainer* [▶ 1890]		Container holding one or more input sample(s) (ContainerType_Vector_REAL, ContainerType_Vector_LREAL, ContainerType_Vector_Vector_REAL, or ContainerType_Vector_Vector_LREAL)
ipScales	ITcVnContainer* [▶ 1890]		Container with the scaling parameters of each feature (type ContainerType_Vector_REAL or ContainerType_Vector_LREAL).
ipScaledSamples	ITcVnContainer* [▶ 1890]&		Returns the scaled sample(s) using the same type like ipSamples. If the same container ipSamples is uesd, the source data will be replaced.
fA	double	0.0	represents the lower bound of the range for TCVN_FST1_MINMAX
fB	double	1.0	represents the upper bound of the range for TCVN_FST1_MINMAX

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.16 FeatureTransform

Apply a feature transform to one or more sample(s).

Syntax

Definition:

```
HRESULT FeatureTransform(
    HRESULT       hrPrev,
    ITcVnMlModel* ipFeatureTransform,
    ITcUnknown*   ipSamples,
    ITcUnknown**  pipTransformed
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipFeatureTransform	ITcVnMlModel* [▶ 1933]	Feature transform instance to be used
ipSamples	ITcUnknown* [▶ 1939]	Container holding one or more of input sample(s) (ContainerType_Vector_REAL, ContainerType_Vector_LREAL, ContainerType_Vector_Vector_REAL, or ContainerType_Vector_Vector_LREAL)
pipTransformed	ITcUnknown* [▶ 1939]*	Returns the transformed sample(s) using the same type like ipSamples

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.17 GetBatchClusters

Get the IDs of the best-matching clusters of a batch of samples.

Syntax

Definition:

```
HRESULT GetBatchClusters(
    HRESULT          hrPrev,
    ITcVnMlModel*    ipClusterer,
    ITcUnknown*      ipSamples,
    ITcVnContainer*& ipClusters,
    ITcVnContainer** pipNovelties = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipClusterer	ITcVnMlModel* [▶ 1933]		Clusterer to be used
ipSamples	ITcUnknown* [▶ 1939]		Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipClusters	ITcVnContainer* [▶ 1890]&		Returns the cluster IDs (ContainerType_Vector_DINT)
pipNovelties	ITcVnContainer* [▶ 1890]*	nullptr	Returns the degree of novelty (0.0 if a sample is completely known; > 0.0 otherwise) of each sample (ContainerType_Vector_REAL; optional, set to 0 if not required)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.18 GetBatchNovelties

Get the degree of novelty of multiple samples.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT GetBatchNovelties(
    HRESULT          hrPrev,
    ITcVnMlModel*    ipNoveltyDetector,
    ITcUnknown*      ipSamples,
    ITcVnContainer*& ipNovelties
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipNoveltyDetector	ITcVnMlModel* [▶ 1933]	Novelty detector to be used
ipSamples	ITcUnknown* [▶ 1939]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipNovelties	ITcVnContainer* [▶ 1890]&	Returns the degree of novelty (0.0 if a sample is completely known; > 0.0 otherwise) of each sample (ContainerType_Vector_REAL)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.19 GetClusterCenter

Get the center of a cluster.

Syntax

Definition:

```
HRESULT GetClusterCenter(
    HRESULT          hrPrev,
    ITcVnMlModel*    ipClusterer,
    LONG             nCluster,
    ITcVnContainer*& ipCenter
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipClusterer	ITcVnMlModel* [▶ 1933]	Clusterer to be used
nCluster	LONG	Cluster ID of the cluster the center of which is requested
ipCenter	ITcVnContainer* [▶ 1890]&	Returns the cluster center (ContainerType_Vector_REAL or ContainerType_Vector_LREAL depending on the floating point type internally used by the clusterer)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.20 GetClusterNum

Get the number of clusters used by this clusterer.

Syntax

Definition:

```
HRESULT GetClusterNum(
    HRESULT       hrPrev,
    ITcVnMlModel* ipClusterer,
    ULONGLONG&    nClusterNum
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipClusterer	ITcVnMlModel* [▶ 1933]	Clusterer to be used
nClusterNum	ULONGLONG&	Returns the number of clusters

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.21 GetFeatureScales

Calculate the scaling parameters for each feature in the input samples based on the scaling type.

Syntax

Definition:

```
HRESULT GetFeatureScales(
    HRESULT                 hrPrev,
    ITcVnContainer*         ipSamples,
    ITcVnContainer*&        ipScales,
    ETcVnFeatureScalingType eFeatureScalingType
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSamples	ITcVnContainer* [▶ 1890]	Container holding input sample(s) (ContainerType_Vector_REAL, ContainerType_Vector_LREAL, ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipScales	ITcVnContainer* [▶ 1890]&	Returns a container with the scaling parameters for each feature (Vector_REAL or Vector_LREAL depending on the type of ipSamples).
eFeatureScalingType	ETcVnFeatureScalingType [▶ 1740]	Feature scaling type

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.22 GetSampleCluster

Get the ID of the best-matching cluster of a single sample.

Syntax

Definition:

```
HRESULT GetSampleCluster(
    HRESULT       hrPrev,
    ITcVnMlModel* ipClusterer,
    ITcUnknown*   ipSample,
    LONG&         nCluster,
    float*        pNovelty = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipClusterer	ITcVnMlModel* [▶ 1933]		Clusterer to be used
ipSample	ITcUnknown* [▶ 1939]		Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
nCluster	LONG&		Returns the ID of the cluster the sample has been assigned to
pNovelty	float*	nullptr	Returns the degree of novelty (0.0 if a sample is completely known; > 0.0 otherwise) of the presented sample (optional, set to 0 if not required)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.23 GetSampleNovelty

Get the degree of novelty of a single sample.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT GetSampleNovelty(
    HRESULT       hrPrev,
    ITcVnMlModel* ipNoveltyDetector,
    ITcUnknown*   ipSample,
    float&        fNovelty
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipNoveltyDetector	ITcVnMlModel* [▶ 1933]	Novelty detector to be used
ipSample	ITcUnknown* [▶ 1939]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
fNovelty	float&	Returns the degree of novelty (0.0 if a sample is completely known; > 0.0 otherwise) of the presented sample

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.24 InverseFeatureScaling

Apply the inverse of feature scaling to one or more sample(s). [fA, fB] should be used if the scaling has been performed using TCVN_FST1_MINMAX with a predetermined output range..

Syntax

Definition:

```
HRESULT InverseFeatureScaling(
    HRESULT          hrPrev,
    ITcVnContainer*  ipSamples,
    ITcVnContainer*  ipScales,
    ITcVnContainer*& ipInverseSamples,
    double           fA = 0.0,
    double           fB = 1.0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSamples	ITcVnContainer* [▶ 1890]		Container holding one or more input sample(s) (ContainerType_Vector_REAL, ContainerType_Vector_LREAL, ContainerType_Vector_Vector_REAL, or ContainerType_Vector_Vector_LREAL)
ipScales	ITcVnContainer* [▶ 1890]		Container with the scaling parameters of each feature (type ContainerType_Vector_REAL or ContainerType_Vector_LREAL).
ipInverseSamples	ITcVnContainer* [▶ 1890]&		Returns the inversed sample(s) using the scaling type and parameters of ipScales. It has the same type of ipSamples. If the same container ipSamples is used, the source data will be replaced.
fA	double	0.0	represents the lower bound of the range for TCVN_FST1_MINMAX
fB	double	1.0	represents the upper bound of the range for TCVN_FST1_MINMAX

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.25 InverseFeatureScaling (float)

Apply the inverse of feature scaling to a single value. [fA, fB] should be used if the scaling has been performed using TCVN_FST1_MINMAX with a predetermined output range.

Syntax

Definition:

```
HRESULT InverseFeatureScaling(
    HRESULT         hrPrev,
    float           fSample,
    ITcVnContainer* ipScales,
    float&          fInverseSample,
    double          fA = 0.0,
    double          fB = 1.0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
fSample	float		Sample value
ipScales	ITcVnContainer* [▶ 1890]		Container with the scaling parameters of each feature (type ContainerType_Vector_REAL or ContainerType_Vector_LREAL).
fInverseSample	float&		Returns the inversed sample.
fA	double	0.0	represents the lower bound of the range for TCVN_FST1_MINMAX
fB	double	1.0	represents the upper bound of the range for TCVN_FST1_MINMAX

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.26 InverseFeatureTransform

Apply an inverse feature transform to one or more sample(s).

Syntax

Definition:

```
HRESULT InverseFeatureTransform(
    HRESULT       hrPrev,
    ITcVnMlModel* ipFeatureTransform,
    ITcUnknown*   ipSamples,
    ITcUnknown**  pipTransformed
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipFeatureTransform	ITcVnMlModel* [▶ 1933]	Feature transform instance to be used
ipSamples	ITcUnknown* [▶ 1939]	Container holding one or more of input sample(s) (ContainerType_Vector_REAL, ContainerType_Vector_LREAL, ContainerType_Vector_Vector_REAL, or ContainerType_Vector_Vector_LREAL)
pipTransformed	ITcUnknown* [▶ 1939]*	Returns the transformed sample(s) using the same type like ipSamples

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.27 PredictBatch

Compute predictions for a batch of samples.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT PredictBatch(
    HRESULT          hrPrev,
    ITcVnMlModel*    ipMlModel,
    ITcUnknown*      ipSamples,
    ITcVnContainer*& ipPredictions,
    ITcVnContainer** pipNovelties = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1933]		Classifier or regressor to be used
ipSamples	ITcUnknown* [▶ 1939]		Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipPredictions	ITcVnContainer* [▶ 1890]&		Returns the predicted outputs (depending on ipSamples; class labels (for classification, ContainerType_Vector_DINT) or real-valued predictions (for regression with scalar output, ContainerType_Vector_REAL or ContainerType_Vector_LREAL; for regression with vectorial output, ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL))
pipNovelties	ITcVnContainer* [▶ 1890]*	nullptr	Returns the degree of novelty (0.0 if a sample is completely known; > 0.0 otherwise) of each sample (ContainerType_Vector_REAL; optional, set to 0 if not required)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.28 PredictSampleClass

Classify a single sample.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT PredictSampleClass(
    HRESULT       hrPrev,
    ITcVnMlModel* ipClassifier,
    ITcUnknown*   ipSample,
    LONG&         nClass,
    float*        pNovelty = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipClassifier	ITcVnMlModel* [▶ 1933]		Classifier to be used
ipSample	ITcUnknown* [▶ 1939]		Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
nClass	LONG&		Returns the classification result
pNovelty	float*	nullptr	Returns the degree of novelty (0.0 if a sample is completely known; > 0.0 otherwise) of the presented sample (optional, set to 0 if not required)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.29 PredictSampleScalar

Compute a scalar prediction for a single sample.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT PredictSampleScalar(
    HRESULT       hrPrev,
    ITcVnMlModel* ipRegressor,
    ITcUnknown*   ipSample,
    float&        fPrediction,
    float*        pNovelty = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipRegressor	ITcVnMlModel* [▶ 1933]		Regressor to be used
ipSample	ITcUnknown* [▶ 1939]		Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
fPrediction	float&		Returns the predicted output
pNovelty	float*	nullptr	Returns the degree of novelty (0.0 if a sample is completely known; > 0.0 otherwise) of the presented sample (optional, set to 0 if not required)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.30 PredictSampleVector

Compute a vectorial prediction for a single sample.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT PredictSampleVector(
    HRESULT          hrPrev,
    ITcVnMlModel*    ipRegressor,
    ITcUnknown*      ipSample,
```

```
    ITcVnContainer*& ipPrediction,
    float*           pNovelty = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipRegressor	ITcVnMlModel* [▶ 1933]		Regressor to be used
ipSample	ITcUnknown* [▶ 1939]		Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
ipPrediction	ITcVnContainer* [▶ 1890]&		Returns the predicted output (ContainerType_Vector_REAL or ContainerType_Vector_LREAL, depending on ipSample)
pNovelty	float*	nullptr	Returns the degree of novelty (0.0 if a sample is completely known; > 0.0 otherwise) of the presented sample (optional, set to 0 if not required)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.31 TrainBatch

Train a classifier, a clusterer, a novelty detector, or a regressor with a batch of samples. On-line trainable models are trained once with each sample. Depending on the application and the number of available training samples, repeated training of such models with the same data may improve the results.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT TrainBatch(
    HRESULT         hrPrev,
    ITcVnMlModel*   ipMlModel,
    ITcUnknown*     ipSamples,
    ITcVnContainer* ipOutputs
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1933]	Classifier, clusterer, novelty detector, or regressor to be used
ipSamples	ITcUnknown* [▶ 1939]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipOutputs	ITcVnContainer* [▶ 1890]	Class labels (for classification, ContainerType_Vector_DINT) or target outputs (for regression with scalar output, ContainerType_Vector_REAL or ContainerType_Vector_LREAL; for regression with vectorial output, ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL) to be learnt (must be 0 for clusterers and for novelty detectors)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.32 TrainBatchClusters

Train a clusterer with a batch of samples and return the IDs of the clusters the samples have been assigned to, if requested. On-line trainable clusterers are trained once with each sample. Depending on the application and the number of available training samples, repeated training of such models with the same data may improve the results.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT TrainBatchClusters(
    HRESULT          hrPrev,
    ITcVnMlModel*    ipClusterer,
    ITcUnknown*      ipSamples,
    ITcVnContainer*& ipClusters
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipClusterer	ITcVnMlModel* [▶ 1933]	Clusterer to be used
ipSamples	ITcUnknown* [▶ 1939]	Container holding a batch of input samples (ContainerType_Vector_Vector_REAL or ContainerType_Vector_Vector_LREAL)
ipClusters	ITcVnContainer* [▶ 1890]&	Returns the IDs of the clusters the samples have been assigned to (ContainerType_Vector_DINT; optional, set to 0 if not required)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.33 TrainSample

Train a clusterer or a novelty detector with a single sample.

Syntax

Definition:

```
HRESULT TrainSample(
    HRESULT       hrPrev,
    ITcVnMlModel* ipMlModel,
    ITcUnknown*   ipSample
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipMlModel	ITcVnMlModel* [▶ 1933]	Clusterer or novelty detector to be used
ipSample	ITcUnknown* [▶ 1939]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.34 TrainSampleClass

Train a classifier with a single sample.

Syntax

Definition:

```
HRESULT TrainSampleClass(
    HRESULT       hrPrev,
    ITcVnMlModel* ipClassifier,
    ITcUnknown*   ipSample,
    LONG          nClass
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipClassifier	ITcVnMlModel* [▶ 1933]	Classifier to be used
ipSample	ITcUnknown* [▶ 1939]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
nClass	LONG	Class label to be learnt

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.35 TrainSampleCluster

Train a clusterer with a single sample and return the ID of the cluster the sample has been assigned to, if requested. The clusterer must be online trainable.

Syntax

Definition:

```
HRESULT TrainSampleCluster(
    HRESULT       hrPrev,
    ITcVnMlModel* ipClusterer,
    ITcUnknown*   ipSample,
    LONG*         pCluster
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipClusterer	ITcVnMlModel* [▶ 1933]	Clusterer to be used
ipSample	ITcUnknown* [▶ 1939]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
pCluster	LONG*	Returns the ID of the cluster the sample has been assigned to (If the pointer is 0, no cluster ID is requested.)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.36 TrainSampleScalar

Train a regressor with a single sample and scalar output.

Syntax

Definition:

```
HRESULT TrainSampleScalar(
    HRESULT       hrPrev,
    ITcVnMlModel* ipRegressor,
    ITcUnknown*   ipSample,
    float         fOutput
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipRegressor	ITcVnMlModel* [▶ 1933]	Regressor to be used
ipSample	ITcUnknown* [▶ 1939]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
fOutput	float	Scalar output to be learnt

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.19.37 TrainSampleVector

Train a regressor with a single sample and vectorial output.

Syntax

Definition:

```
HRESULT TrainSampleVector(
    HRESULT         hrPrev,
    ITcVnMlModel*   ipRegressor,
    ITcUnknown*     ipSample,
    ITcVnContainer* ipOutput
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipRegressor	ITcVnMlModel* [▶ 1933]	Regressor to be used
ipSample	ITcUnknown* [▶ 1939]	Container holding a single input sample (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)
ipOutput	ITcVnContainer* [▶ 1890]	Vectorial output to be learnt (ContainerType_Vector_REAL or ContainerType_Vector_LREAL)

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.20 Measurement

Die Gruppe enthält Funktionen zur Vermessung von Objekten auf Bildern.

Funktionen

Kantenlokalisierung

- LocateCircularArc [▶ 2658] zur Lokalisierung von anteiligen Kreisbögen
- LocateEdge [▶ 2661] zur Lokalisierung von linearen Kanten
- LocateEdges [▶ 2665] zur Lokalisierung von mehreren linearen Kanten
- LocateEllipse [▶ 2668] zur Lokalisierung von vollständigen Ellipsen

Geometrische Merkmale

- ClosestPointsBF [▶ 2657] zur Berechnung der nächstgelegenen Punkte zweier Punktewolken
- MeasureAngleBetweenEdges [▶ 2671] zur Vermessung des Winkels zwischen zwei Kanten
- MeasureEdgeDistance [▶ 2674] zur Vermessung des durchschnittlichen Abstands zweier Kanten
- MeasureMinEdgeDistance [▶ 2678] zur Vermessung des minimalen Abstands zweier Kanten

Sonstiges

- AdjustSearchWindowOrientationToLinearEdge [▶ 2656] zur Ausrichtung des Suchfelds zu einer Kante

Parameter zur Kantenlokalisierung

Die Grenzen eines Bildobjekts liegen häufig nicht exakt auf der Grenze zwischen zwei Pixeln. Sie verlaufen stattdessen unabhängig von Pixelraster. Daher ist eine Subpixel-genaue Kantenlokalisierung nötig. Diese betrachtet nicht nur den Übergang zwischen zwei Pixeln, sondern bestimmt anhand des gesamten Intensitätsverlaufs einer Objektgrenze die Kantenposition innerhalb eines Pixels.

HINWEIS

Bild vor Vermessung nicht bearbeiten

Bearbeiten Sie das Bild vor dem Aufruf der Vermessungsfunktion nicht. Eine Bearbeitung kann zu Informationsverlust und entsprechend geringerer Genauigkeit führen.

Diese Grundfunktionalität der Kantenlokalisierung wird von allen Vermessungsfunktionen verwendet. Die Funktionen unterscheiden sich hauptsächlich darin, in welcher geometrischen Anordnung sie die Kantenlokalisierung durchführen (z. B. LocateEllipse [▶ 2668] für runde Objekte) und welche Eigenschaften Sie daraus berechnen (z. B. MeasureAngleBetweenEdges [▶ 2671] zur Winkelvermessung).

Bei jeder Funktion, die Kantenlokalisierung nutzt, müssen bestimmte Parameter definiert werden. Dazu gehören:

- Definition des Suchbereichs: Wo wird die Kante gesucht?
- Definition des Kantenkontrasts: Was wird als Kante erkannt?
- Definition des Vermessungsalgorithmus: Wie soll die genaue Position der Kante bestimmt werden.

Diese Parameter sind folgend beschrieben. Bei Expert-Parametern sind zudem Standard-Werte angegeben.

Suchbereich

Parameter	Beschreibung
Lage der Suchlinien	Die Lage der Suchlinien wird je nach zu lokalisierendem Merkmal definiert. Die Lage von Suchlinien für eine lineare Kante wird als einzelne Linie mit Startpunkt `aStartPoint` und Endpunkt `aEndPoint` definiert. Die Lage von Suchlinien für eine Ellipse wird durch einen Mittelpunkt `aCenter` und einen Radius `fSearchRadius` definiert. Die Lage von Suchlinien für Winkel wird durch einen Innenpunkt `aInnerPoint` und zwei Außenpunkte `aOuterPoint1` und `aOuterPoint2` definiert. Die Lage der Suchlinien sollte immer so eingestellt werden, dass die Suchlinien die zu lokalisierenden Kanten orthogonal schneiden. Ein rechteckiger Suchbereich kann bei Bedarf mit der Funktion AdjustSearchWindowOrientationToLinearEdge [▶ 2656] orthogonal an eine Kante angepasst werden.
Anzahl der Suchlinien `nSearchLines`	Hiermit wird festgelegt, wie viele Suchlinien für die Kantenlokalisierung verwendet werden sollen. Die einfachste Kantenlokalisierung hat eine Suchlinie, auf der die Kante gefunden werden muss. Bei mehreren Suchlinien werden je nach zu bestimmendem geometrischem Merkmal örtlich verteilt, sodass mehrere Kantenübergänge lokalisiert werden. Dies kann in folgenden Fällen nötig sein: • Nicht nur die Position, sondern auch der Verlauf oder die Rotation der Kante gefragt sind. • Das geometrische Merkmal (z. B. Ellipse) erfordert mehrere Suchlinien. • Ein Durchschnittswert ist gefordert. Bei der Lokalisierung von linearen Kanten werden die Suchlinien symmetrisch auf beiden Seiten der zentralen Suchlinie angeordnet, wobei die zentrale Suchlinie in `nSearchLines` inklusiv ist. Somit wird immer eine ungerade Anzahl an Suchlinien benötigt.
Abstand der Suchlinien `fSearchlineDist` *(Expert)* Standard: `1`	Die Suchlinien-Distanz definiert den Abstand zwischen benachbarten Suchlinien in Pixeln (> 0, default 1). Dadurch kann flexibel zwischen Genauigkeit und benötigter Rechenzeit abgewogen werden.
Invertierung der Suchrichtung `bInvertSearchDirection` *(Expert)* Standard: `false`	Der Lokalisierungsalgorithmus sucht die Kante immer von einem Ende der Suchlinie zum anderen. Z. B. wird von aStartPoint nach aEndPoint gesucht. Wenn es aufgrund der Bildbeschaffenheit nötig ist diese Suchrichtung zu invertieren (also vom anderen Ende der Suchlinien aus anzufangen), hilft dieser Parameter.

Kontrast der Kante

Parameter	Beschreibung
Kantenrichtung `eEdgeDirection`	Die Kantenrichtung `ETcVnEdgeDirection` legt fest, ob ein Kantenübergang von Hell nach Dunkel oder von Dunkel nach Hell gefunden werden soll. Dazu stehen folgende Werte zur Verfügung: • `ED_DARK_TO_LIGHT` • `ED_LIGHT_TO_DARK`
Minimalstärke `fMinStrength`	Die Minimalstärke legt fest, wie groß der Intensitätsunterschied innerhalb einer Kante mindestens sein muss, damit sie gefunden wird. Dadurch lässt sich steuern, wie viel Kontrast eine Kante haben muss, um gefunden zu werden.
Maximaldicke `nMaxThickness` *(Expert)* Standard: `10`	Die Maximaldicke legt fest, innerhalb von wie vielen Pixeln `fMinStrength` erreicht werden muss. Dadurch lässt sich steuern, ob nur scharfe Kanten oder auch sehr unscharfe Kanten gefunden werden sollen.

Algorithmus

Parameter	Beschreibung
Algorithmus `eAlgorithm` *(Expert)* Standard: `EDA_INTERPOLATION`	Der Parameter `ETcVnEdgeDetectionAlgorithm` definiert den verwendeten Lokalisierungsalgorithmus. Zur Verfügung stehen: • `EDA_INTERPOLATION` • `EDA_APPROX_ERF` • `EDA_APPROX_GAUSSIAN` `EDA_APPROX_ERF` und `EDA_APPROX_GAUSSIAN` sind langsamer als der Standard-Algorithmus `EDA_INTERPOLATION`, können aber eine höhere Genauigkeit erreichen. Voraussetzung dafür ist allerdings, dass der Intensitätsverlauf der Kante auch möglichst gut dem vorgegebenen Modell entspricht. Als Modell werden die Fehlerfunktion `erf` bzw. die Gauß-Funktion verwendet. Die `erf`-Funktion ist gut für Kanten geeignet, wenn sich der Übergang von hell nach dunkel über mehrere Pixel erstreckt und auf beiden Seiten der helle bzw. dunkle Bereich in etwa konstant bleibt. Eine Gauß-Funktion ist gut für Linien geeignet, d. h. ein Übergang dunkel-hell-dunkel oder umgekehrt. Das Helligkeitsplateau darf dabei aber nur wenige Pixel breit sein, ansonsten sollte die erf-Funktion verwendet werden.
Subpixel / Iterationen `nSubpixelsIterations` *(Expert)* Standard: 10	Die Bedeutung dieses Parameters ist abhängig vom verwendeten Lokalisierungsalgorithmus. Bei `EDA_INTERPOLATION`: Die Anzahl der betrachteten Subpixel definiert, wie feingranular die Suchlinie die unterliegenden Pixelintensitäten auswertet. Ein Wert von 10 gibt z. B. an, dass auf der Länge von 1 Pixel 10 Messwerte genommen werden. Empfohlen: `5 - 10` Bei `EDA_APPROX_ERF` und `EDA_APPROX_GAUSSIAN`: Die Anzahl der Iterationen ist ein Abbruchkriterium, das definiert, mit maximal wie vielen Iterationen das Kantenmodell auf die Pixelintensitäten approximiert werden sollen. Mehr Iterationen erhöhen die Laufzeit, wenn diese zu hoch ist sollte die Interpolationsmethode verwendet werden. Empfohlen: `50 - 100`
Näherungsgenauigkeit `fApproxPrecision` *(Expert)* Standard: `0.001`	Die Näherungsgenauigkeit ist ein weiteres Abbruchkriterium der Kantenlokalisierung. Sie wird nur beachtet, wenn der Lokalisierungsalgorithmus `EDA_APPROX_ERF` oder `EDA_APPROX_GAUSSIAN` ist.

Parameter	Beschreibung
	Die Approximation der Kante bzw. der Linie wird abgebrochen, wenn entweder die Anzahl der Iterationen von `nSubPixelIterations` erreicht oder die relative Abweichung in Pixeln zwei aufeinanderfolgender Iterationen kleiner als `fApproximationPrecision` ist Empfohlen: `0.01` - `0.0001`

Interpretation des HRESULT

Für Measurement-Funktionen haben die HRESULT [▶ 150]–Rückgabewerte folgende Bedeutung:

Code	Name	Beschreibung
0x000	S_OK	Funktion wurde erfolgreich ausgeführt und auf allen Suchlinien wurde eine Kante lokalisiert.
0x001	S_FALSE	Funktion wurde erfolgreich ausgeführt, aber es wurde nicht auf jeder Suchlinie eine Kante lokalisiert.
0x256	S_WATCHDOGTIMEOUT	Funktion wurde vom Watchdog abgebrochen. In diesem Fall können alle Measurement-Funktionen Teilergebnisse zurückgeben.
0x70C	NOTFOUND	Funktion wurde erfolgreich ausgeführt, aber auf keiner Suchlinie konnte eine Kante gefunden werden. Es wird davon ausgegangen, dass eine Vermessungsfunktion nur aufgerufen wird, wenn sichergestellt ist, dass das Bild ein Objekt mit den parametrierten Kanteneigenschaften enthält. Ein Nicht-Finden aller Suchlinien deutet dabei auf eine falsche Parametrierung oder einen ungeeigneten Aufruf der Funktion hin. Daher wird in diesem Fall ein Error-Code ausgegeben.
0x7xx	Alle Error-Codes außer NOTFOUND	Funktion wurde nicht erfolgreich ausgeführt. Siehe dazu: ADS Return Codes [▶ 2966]

Zur Verarbeitung des Rückgabewerts kann beispielhaft folgende Abfrage benutzt werden:

```
if (SUCCEEDED(hr))
{
    // Process results
    if (hr != S_OK)
    {
        // Emit warning that not all search lines were found
    }
}
else if (hr == ADS_E_NOTFOUND)
{
    // Emit warning for wrong parametrization or missing piece
}
else
{
    // Error handling
}
```

Ausführungszeit reduzieren

Die Ausführungszeit schwankt bei einem festen Suchfenster leicht je nach Position des Bauteils im Bild. Ist die gefundene Kante näher am Startpunkt, ist die Ausführungszeit kürzer als bei einer weiter entfernten Kante.

Ein größeres Schwanken kann bei den beiden Approximierungs-Algorithmen auftreten, da diese unterschiedlich viele Iterationen benötigen, um die Modellparameter mittels der umliegenden Pixelintensitäten zu approximieren.

Generell werden weniger Iterationen benötigt, wenn die Suchlinien des Suchfensters möglichst orthogonal auf die Kante treffen. Falls die Position und Orientierung der Objekte im Bild keinen oder nur geringen Schwankungen unterliegen, richten Sie das Suchfenster entsprechend aus, um kürzere Ausführungszeiten zu erreichen. Zudem lässt sich die maximal benötigte Zeit reduzieren, indem die maximale Anzahl an Iterationen verringert wird, was aber ggf. auch ungenauere Ergebnisse liefert.

Zudem kann die Zeit auch mit einem Watchdog von außen begrenzt werden, der die Ausführung der Funktion bei Bedarf abbricht und die vorhandenen Teilergebnisse zurückliefert.

6.2.3.20.1 AdjustSearchWindowOrientationToLinearEdge

Adjust the search window to be orthogonal to a linear edge. Might be helpful before calling edge localization and distance measurement functions that rely on the search window.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT AdjustSearchWindowOrientationToLinearEdge(
    HRESULT            hrPrev,
    ITcVnImage*        ipSrcImage,
    TcVnPoint2_REAL&   aStartPoint,
    TcVnPoint2_REAL&   aEndPoint,
    ETcVnEdgeDirection eEdgeDirection,
    float              fMinStrength,
    ULONG              nMaxThickness,
    ULONG              nSearchLines
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
aStartPoint	TcVnPoint2_REAL [▶ 1711]&	Position from which to start the search process (in the direction of aEndPoint), which is not changed
aEndPoint	TcVnPoint2_REAL [▶ 1711]&	Position where the search process ends, which is adjusted by this function
eEdgeDirection	ETcVnEdgeDirection [▶ 1738]	Specification of the edge direction to search for
fMinStrength	float	Specification of the minimum strength (intensity difference) of the edge to search for
nMaxThickness	ULONG	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels
nSearchLines	ULONG	Width of the search window, i.e. the number of search lines (3, 5, 7, ...), centered around the line specified by aStartPoint and aEndPoint

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.20.2 ClosestPointsBF

Find the closest distance between two 2d point sets of the same type, using a brute force approach.

Syntax

Definition:

```
HRESULT ClosestPointsBF(
    HRESULT           hrPrev,
    ITcVnContainer*   ipPointSet1,
    ITcVnContainer*   ipPointSet2,
    double&           fMinDist,
    TcVnPoint2_LREAL& aPoint1,
    TcVnPoint2_LREAL& aPoint2
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipPointSet1	ITcVnContainer* [▶ 1890]	Container with 1st set of 2d points
ipPointSet2	ITcVnContainer* [▶ 1890]	Container with 2nd set of 2d points
fMinDist	double&	Returns the minimum distance between 2 points of the different point sets
aPoint1	TcVnPoint2_LREAL [▶ 1711]&	Returns the point out of ipPointSet1, for which fMinDist is achieved
aPoint2	TcVnPoint2_LREAL [▶ 1711]&	Returns the point out of ipPointSet2, for which fMinDist is achieved

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.20.3 LocateCircularArc

Locate a circular arc.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT LocateCircularArc(
    HRESULT                     hrPrev,
    ITcVnImage*                 ipSrcImage,
    TcVnCircularArc&            stCircularArc,
    TcVnPoint2_REAL&            aCenter,
    float                       fSearchRadius,
    double                      fArcDirectionRad,
    ETcVnEdgeDirection          eEdgeDirection,
    float                       fMinStrength,
    double                      fAngleStepRad = 0.1,
    ULONG                       nMaxThickness = 10,
    bool                        bInvertSearchDirection = false,
    ULONG                       nSubpixelsIterations = 10,
    float                       fApproxPrecision = 0.001f,
    ETcVnEdgeDetectionAlgorithm eAlgorithm = EDA_INTERPOLATION,
    ITcVnContainer**            pipContourPoints = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (1 channel)
stCircularArc	TcVnCircularArc [▶ 1765]&		Returns the circular arc parameters
aCenter	TcVnPoint2_REAL [▶ 1711]&		Input estimated circle center (only used as a starting point to search for the circle contour, not used for the circle center estimation)
fSearchRadius	float		Input search radius (starting from aCenter, should be greater than the actual circle radius but aCenter + fSearchRadius should be within the image borders)
fArcDirectionRad	double		Input search starting direction in radian. Valid range is [-pi, +pi], where right is 0 rad, top is -pi/2 rad and bottom +pi/2 rad. The circular arc should at least be valid in range fArcDirectionRad +- 4 * fAngleStepRad)
eEdgeDirection	ETcVnEdgeDirection [▶ 1738]		Specification of the edge direction to search for
fMinStrength	float		Specification of the minimum strength (intensity difference) of the edge to search for
fAngleStepRad	double	0.1	Search step in rad (should be chosen so that about 20 - 60 steps are available for the whole arc. In most cases 0.1 rad (5.7 deg) is a good value)
nMaxThickness	ULONG	10	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels
bInvertSearchDirection	bool	false	If true, the search starts from outside the circular arc in direction of the center

Name	Type	Default	Description
nSubpixelsIterations	ULONG	10	Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)
fApproxPrecision	float	0.001f	Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)
eAlgorithm	ETcVnEdgeDetectionAlgorithm [▶ 1738]	EDA_INTERPOLATION	Selection of the edge detection algorithm
pipContourPoints	ITcVnContainer* [▶ 1890]*	nullptr	Returns the subpixel accurate contour (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_REAL; Non-zero interface pointers are reused.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.20.4 LocateEdge

Locate the points of the first occuring edge inside a specified search window with subpixel accuracy.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT LocateEdge(
    HRESULT                 hrPrev,
    ITcVnImage*             ipSrcImage,
    ITcVnContainer*&        ipEdgePoints,
    TcVnPoint2_REAL&        aStartPoint,
    TcVnPoint2_REAL&        aEndPoint,
    ETcVnEdgeDirection      eEdgeDirection,
    float                   fMinStrength,
```

```
    ULONG                          nSearchLines,
    float&                         fAvgStrength,
    float                          fSearchLineDist = 1,
    ULONG                          nMaxThickness = 10,
    ULONG                          nSubpixelsIterations = 10,
    float                          fApproxPrecision = 0.001f,
    ETcVnEdgeDetectionAlgorithm eAlgorithm = EDA_INTERPOLATION
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (1 channel)
ipEdgePoints	ITcVnContainer* [▶ 1890]&		Returns the detected edge points (ContainerType_Vector_TcVnPoint2_REAL)
aStartPoint	TcVnPoint2_REAL [▶ 1711]&		Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 1711]&		Position where the search process ends
eEdgeDirection	ETcVnEdgeDirection [▶ 1738]		Specification of the edge direction to search for
fMinStrength	float		Specification of the minimum strength (intensity difference) of the edge to search for
nSearchLines	ULONG		Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ...), centered around the line specified by aStartPoint and aEndPoint
fAvgStrength	float&		Returns the average strength of the detected edge
fSearchLineDist	float	1	Distance between the search lines in pixels (> 0)
nMaxThickness	ULONG	10	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels
nSubpixelsIterations	ULONG	10	Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)

Name	Type	Default	Description
fApproxPrecision	float	0.001f	Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)
eAlgorithm	ETcVnEdgeDetectionAlgorithm [▶ 1738]	EDA_INTERPOLATION	Selection of the edge detection algorithm

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.20.5 LocateEdges

Locate the points of multiple occuring edges inside a specified search window with subpixel accuracy.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT LocateEdges(
    HRESULT                     hrPrev,
    ITcVnImage*                 ipSrcImage,
    ITcVnContainer*&            ipEdgePoints,
    TcVnPoint2_REAL&            aStartPoint,
    TcVnPoint2_REAL&            aEndPoint,
    ULONG                       nNumEdges,
    ETcVnEdgeDirection          eEdgeDirection,
    bool                        bAlternateDirection,
    float                       fMinStrength,
    ULONG                       nSearchLines,
    float&                      fAvgStrength,
    float                       fSearchLineDist = 1,
    ULONG                       nMaxThickness = 10,
    ULONG                       nSubpixelsIterations = 10,
    float                       fApproxPrecision = 0.001f,
    ETcVnEdgeDetectionAlgorithm eAlgorithm = EDA_INTERPOLATION
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (1 channel)
ipEdgePoints	ITcVnContainer* [▶ 1890]&		Returns the detected edge points (ContainerType_Vector_Vector_TcVnPoint2_REAL)
aStartPoint	TcVnPoint2_REAL [▶ 1711]&		Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 1711]&		Position where the search process ends
nNumEdges	ULONG		The (maximum) number of edges to search for
eEdgeDirection	ETcVnEdgeDirection [▶ 1738]		Specification of the edge direction to search for
bAlternateDirection	bool		If true, eEdgeDirection is alternated after each detected edge. Else, only edges with eEdgeDirection are searched for.
fMinStrength	float		Specification of the minimum strength (intensity difference) of the edge to search for
nSearchLines	ULONG		Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ...), centered around the line specified by aStartPoint and aEndPoint
fAvgStrength	float&		Returns the average strength of the detected edges
fSearchLineDist	float	1	Distance between the search lines in pixels (> 0)
nMaxThickness	ULONG	10	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels

Name	Type	Default	Description
nSubpixelsIterations	ULONG	10	Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)
fApproxPrecision	float	0.001f	Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)
eAlgorithm	ETcVnEdgeDetectionAlgorithm [▶ 1738]	EDA_INTERPOLATION	Selection of the edge detection algorithm

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.20.6 LocateEllipse

Locate an ellipse.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT LocateEllipse(
    HRESULT                     hrPrev,
    ITcVnImage*                 ipSrcImage,
    TcVnRotatedRectangle&       stEllipse,
    TcVnPoint2_REAL&            aCenter,
    float                       fSearchRadius,
    ETcVnEdgeDirection          eEdgeDirection,
    float                       fMinStrength,
    ULONG                       nMaxThickness = 10,
    bool                        bInvertSearchDirection = false,
    float                       fMinSearchRadius = 0,
    ULONG                       nSubpixelsIterations = 10,
    ULONG                       nSearchLines = 92,
    float                       fApproxPrecision = 0.001f,
    ETcVnEdgeDetectionAlgorithm eAlgorithm = EDA_INTERPOLATION,
    ITcVnContainer**            pipContourPoints = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (1 channel)
stEllipse	TcVnRotatedRectangle [▶ 1782]&		Returns the detected ellipse
aCenter	TcVnPoint2_REAL [▶ 1711]&		The expected ellipse center
fSearchRadius	float		Radius around aCenter to search for edges
eEdgeDirection	ETcVnEdgeDirection [▶ 1738]		Specification of the edge direction to search for (from center to outside ellipse or other way round, if bInvertSearchDirection is true)
fMinStrength	float		Specification of the minimum strength (intensity difference) of the edge to search for
nMaxThickness	ULONG	10	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels
bInvertSearchDirection	bool	false	If true, the search starts from outside the ellipse in direction of the center
fMinSearchRadius	float	0	Radius around aCenter to skip before starting to search for edges (e.g. to save time or if the center contains edges that should be ignored)
nSubpixelsIterations	ULONG	10	Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)
nSearchLines	ULONG	92	Specifies the amount of search lines, which are equally distributed in all directions (must be >= 8 and a multiple of 4)

Name	Type	Default	Description
fApproxPrecision	float	0.001f	Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)
eAlgorithm	ETcVnEdgeDetectionAlgorithm [▶ 1738]	EDA_INTERPOLATION	Selection of the edge detection algorithm
pipContourPoints	ITcVnContainer* [▶ 1890]*	nullptr	Returns the subpixel accurate ellipse contour (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_REAL; Non-zero interface pointers are reused.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.20.7 MeasureAngleBetweenEdges

Measure the angle between 2 edges.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT MeasureAngleBetweenEdges(
    HRESULT                     hrPrev,
    ITcVnImage*                 ipSrcImage,
    float&                      fAngle,
    TcVnPoint2_REAL&            aInnerPoint,
    TcVnPoint2_REAL&            aOuterPoint1,
    TcVnPoint2_REAL&            aOuterPoint2,
    ETcVnEdgeDirection          eEdgeDirection,
    float                       fMinStrength,
    ULONG                       nSearchLines,
    float                       fSearchLineDist = 1,
    ULONG                       nMaxThickness = 10,
    bool                        bInvertSearchDirection = false,
    ULONG                       nSubpixelsIterations = 10,
    bool                        bAngleInDegrees = false,
    float                       fApproxPrecision = 0.001f,
    ETcVnEdgeDetectionAlgorithm eAlgorithm = EDA_INTERPOLATION,
    ITcVnContainer**            pipEdgePoints1 = nullptr,
    ITcVnContainer**            pipEdgePoints2 = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (1 channel)
fAngle	float&		Returns the angle between the detected edges (radians or degrees, depending on bAngleInDegrees)
aInnerPoint	TcVnPoint2_REAL [▶ 1711]&		Position inside the angle to measure, from which to start the edge search process (in the direction of aOuterPoint1 and aOuterPoint2)
aOuterPoint1	TcVnPoint2_REAL [▶ 1711]&		Position where the search process for edge 1 ends
aOuterPoint2	TcVnPoint2_REAL [▶ 1711]&		Position where the search process for edge 2 ends
eEdgeDirection	ETcVnEdgeDirection [▶ 1738]		Specification of the edge direction from aInnerPoint to aOuterPoint1 and aOuterPoint2 to search for
fMinStrength	float		Specification of the minimum strength (intensity difference) of the edge to search for
nSearchLines	ULONG		Width of the search window, i.e. the number of search lines (3, 5, 7, ...), centered around the line specified by aInnerPoint and aOuterPoint1, aOuterPoint2
fSearchLineDist	float	1	Distance between the search lines in pixels (> 0)
nMaxThickness	ULONG	10	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels
bInvertSearchDirection	bool	false	If true, the search starts from each aOuterPoint in direction of aInnerPoint

Name	Type	Default	Description
nSubpixelsIterations	ULONG	10	Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)
bAngleInDegrees	bool	false	fAngle is in degrees, if true
fApproxPrecision	float	0.001f	Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)
eAlgorithm	ETcVnEdgeDetectionAlgorithm [▶ 1738]	EDA_INTERPOLATION	Selection of the edge detection algorithm
pipEdgePoints1	ITcVnContainer* [▶ 1890]*	nullptr	Returns the detected edge points between aInnerPoint and aOuterPoint1 (ContainerType_Vector_TcVnPoint2_REAL)
pipEdgePoints2	ITcVnContainer* [▶ 1890]*	nullptr	Returns the detected edge points between aInnerPoint and aOuterPoint2 (ContainerType_Vector_TcVnPoint2_REAL)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.20.8 MeasureEdgeDistance

Measure the distance between 2 parallel edges.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT MeasureEdgeDistance(
    HRESULT                       hrPrev,
    ITcVnImage*                   ipSrcImage,
    float&                        fAvgDistance,
    TcVnPoint2_REAL&              aStartPoint,
    TcVnPoint2_REAL&              aEndPoint,
    ETcVnEdgeDirection            eEdgeDirection,
    float                         fMinStrength,
    ULONG                         nSearchLines,
    float                         fSearchLineDist = 1,
    ULONG                         nMaxThickness = 10,
    bool                          bInvertSearchDirection = false,
    float                         fSearchGap = 0,
    ULONG                         nSubpixelsIterations = 10,
    float                         fApproxPrecision = 0.001f,
    ETcVnEdgeDetectionAlgorithm eAlgorithm = EDA_INTERPOLATION,
    ITcVnContainer**              pipEdgePoints1 = nullptr,
    ITcVnContainer**              pipEdgePoints2 = nullptr,
    ITcVnContainer**              pipDistances = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (1 channel)
fAvgDistance	float&		Returns the average distance between the detected edges
aStartPoint	TcVnPoint2_REAL [▶ 1711]&		Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 1711]&		Position where the search process ends
eEdgeDirection	ETcVnEdgeDirection [▶ 1738]		Specification of the edge direction from aStartPoint to aEndPoint to search for
fMinStrength	float		Specification of the minimum strength (intensity difference) of the edge to search for
nSearchLines	ULONG		Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ...), centered around the line specified by aStartPoint and aEndPoint
fSearchLineDist	float	1	Distance between the search lines in pixels (> 0)
nMaxThickness	ULONG	10	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels
bInvertSearchDirection	bool	false	If true, the search starts from the center point between aStartPoint and aEndPoint in both directions
fSearchGap	float	0	Optional width of a gap (>= 0, centered between aStartPoint and aEndPoint), that is neglected for searching edges (can reduce execution time). The 2 edges to search for must be on different sides of the gap.

Name	Type	Default	Description
nSubpixelsIterations	ULONG	10	Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)
fApproxPrecision	float	0.001f	Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)
eAlgorithm	ETcVnEdgeDetectionAlgorithm [▶ 1738]	EDA_INTERPOLATION	Selection of the edge detection algorithm
pipEdgePoints1	ITcVnContainer* [▶ 1890]*	nullptr	Returns the detected edge points of the edge near aStartPoint (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_REAL)
pipEdgePoints2	ITcVnContainer* [▶ 1890]*	nullptr	Returns the detected edge points of the edge near aEndPoint (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_REAL)
pipDistances	ITcVnContainer* [▶ 1890]*	nullptr	Returns the distances between the detected edge points (optional, set to 0 if not required; ContainerType_Vector_REAL)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.20.9 MeasureMinEdgeDistance

Measure the minimum distance within the specified search window between 2 edges.
Can use available TwinCAT Job Tasks for executing parallel code regions.
Can return partial results when canceled by Watchdog.

Syntax

Definition:

```
HRESULT MeasureMinEdgeDistance(
    HRESULT                       hrPrev,
    ITcVnImage*                   ipSrcImage,
    float&                        fMinDistance,
    TcVnPoint2_REAL&              aStartPoint,
    TcVnPoint2_REAL&              aEndPoint,
    ETcVnEdgeDirection            eEdgeDirection,
    float                         fMinStrength,
    ULONG                         nSearchLines,
    TcVnPoint2_REAL&              aPoint1,
    TcVnPoint2_REAL&              aPoint2,
    float                         fSearchLineDist = 1,
    ULONG                         nMaxThickness = 10,
    bool                          bInvertSearchDirection = false,
    float                         fSearchGap = 0,
    ULONG                         nSubpixelsIterations = 10,
    float                         fApproxPrecision = 0.001f,
    ETcVnEdgeDetectionAlgorithm eAlgorithm = EDA_INTERPOLATION,
    ITcVnContainer**              pipEdgePoints1 = nullptr,
    ITcVnContainer**              pipEdgePoints2 = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (1 channel)
fMinDistance	float&		Returns the minimum distance between the detected edges
aStartPoint	TcVnPoint2_REAL [▶ 1711]&		Position from which to start the search process (in the direction of aEndPoint)
aEndPoint	TcVnPoint2_REAL [▶ 1711]&		Position where the search process ends
eEdgeDirection	ETcVnEdgeDirection [▶ 1738]		Specification of the edge direction from aStartPoint to aEndPoint to search for
fMinStrength	float		Specification of the minimum strength (intensity difference) of the edge to search for
nSearchLines	ULONG		Width of the search window, i.e. the number of search lines (1, 3, 5, 7, ...), centered around the line specified by aStartPoint and aEndPoint
aPoint1	TcVnPoint2_REAL [▶ 1711]&		Returns the point on the edge near aStartPoint, for which the minimum distance is achieved
aPoint2	TcVnPoint2_REAL [▶ 1711]&		Returns the point on the edge near aEndPoint, for which the minimum distance is achieved
fSearchLineDist	float	1	Distance between the search lines in pixels (> 0)
nMaxThickness	ULONG	10	Specification of the maximum thickness of the edge to search for, which means fMinStrength must be reached within nMaxThickness pixels
bInvertSearchDirection	bool	false	If true, the search starts from the center point between aStartPoint and aEndPoint in both directions

Name	Type	Default	Description
fSearchGap	float	0	Optional width of a gap (>= 0, centered between aStartPoint and aEndPoint), that is neglected for searching edges (can reduce execution time). The 2 edges to search for must be on different sides of the gap.
nSubpixelsIterations	ULONG	10	Specifies the number of subpixels (for INTERPOLATION, 10 - 20 usually is sufficient) or maximum number of iterations for optimizing the parameters (for APPROX_ERF and APPROX_GAUSSIAN, 50 - 100 usually is sufficient)
fApproxPrecision	float	0.001f	Specifies the approximation precision for APPROX_ERF and APPROX_GAUSSIAN (0.001 usually is sufficient, unused for INTERPOLATION)
eAlgorithm	ETcVnEdgeDetectionAlgorithm [▶ 1738]	EDA_INTERPOLATION	Selection of the edge detection algorithm
pipEdgePoints1	ITcVnContainer* [▶ 1890]*	nullptr	Returns the detected edge points of the edge near aStartPoint (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_REAL)
pipEdgePoints2	ITcVnContainer* [▶ 1890]*	nullptr	Returns the detected edge points of the edge near aEndPoint (optional, set to 0 if not required; ContainerType_Vector_TcVnPoint2_REAL)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Metrology 2D

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21 Miscellaneous

Diese Gruppe enthält weitere Funktionen, die Gruppenübergreifend verwendet werden können.

Funktionen

Geometrische Hilfsfunktionen

- ClipLineToBoundary [▶ 2684]
- ClipLineToBoundary_ITcVnImage [▶ 2685]
- ClipLineToBoundary_TcVnRectangle_DINT [▶ 2686]
- LineIntersectionPoint [▶ 2693]
- LineIntersectionPointAndAngle [▶ 2694]
- RotatedRectangleCorners [▶ 2696]
- RotatedRectangleIntersection [▶ 2697]
- OrthogonalVector_TcVnVector2_DINT [▶ 2699]
- OrthogonalVector_TcVnVector2_LREAL [▶ 2699]
- OrthogonalVector_TcVnVector2_REAL [▶ 2700]
- PointToLineDistance_TcVnPoint2_DINT [▶ 2701]
- PointToLineDistance_TcVnPoint2_LREAL [▶ 2702]
- PointToLineDistance_TcVnPoint2_REAL [▶ 2703]
- PointToPointDirection_TcVnPoint2_DINT [▶ 2703]
- PointToPointDirection_TcVnPoint2_LREAL [▶ 2704]
- PointToPointDirection_TcVnPoint2_REAL [▶ 2705]
- PointToPointDistance_TcVnPoint2_DINT [▶ 2706]
- PointToPointDistance_TcVnPoint2_LREAL [▶ 2707]
- PointToPointDistance_TcVnPoint2_REAL [▶ 2707]

Zeitstempel

- GetTimestamp [▶ 2690]
- UpdateTimestamp [▶ 2698]

Hilfsfunktionen für Matrizen

- InitMatrixStruct [▶ 2692]
- InvertMatrix3x3 [▶ 2692]
- MultiplyMatrices [▶ 2695]

Konvertieren von Interfaces

- ConvertITcUnknownToITcVnBitmapExport [▶ 2687]
- ConvertITcUnknownToITcVnContainer [▶ 2688]
- ConvertITcUnknownToITcVnImage [▶ 2688]
- ConvertITcUnknownToITcVnMlModel [▶ 2689]

Funktion Initialisierung

- CheckFunctionInitialization [▶ 2684]
- DeinitializeFunction [▶ 2690]

Sonstiges

- HuMomentInvariants [▶ 2691]
- SetRngSeed [▶ 2697]

6.2.3.21.1 CheckFunctionInitialization

Check if a function is initialized with the specified options.

Syntax

Definition:

```
HRESULT CheckFunctionInitialization(
    HRESULT                    hrPrev,
    ETcVnInitializableFunction eFunction,
    ULONGLONG                  nOptions
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
eFunction	ETcVnInitializableFunction [▶ 1743]	Initializable function
nOptions	ULONGLONG	Initialization options for the function

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.2 ClipLineToBoundary

Clips a line to a rectangular boundary.

Syntax

Definition:

```
HRESULT ClipLineToBoundary(
    HRESULT            hrPrev,
    TcVnVector4_LREAL& aLine,
    LONG               nX1,
    LONG               nY1,
    LONG               nX2,
    LONG               nY2,
    TcVnPoint2_REAL&   aStartPoint,
    TcVnPoint2_REAL&   aEndPoint
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aLine	TcVnVector4_LREAL [▶ 1713]&	The first and second element describe the x and y component of the direction vector. The third and fourth element describe the x and y component of the position vector.
nX1	LONG	x of top left rectangle point
nY1	LONG	y of top left rectangle point
nX2	LONG	x of bottom right rectangle point
nY2	LONG	Y of bottom right rectangle point
aStartPoint	TcVnPoint2_REAL [▶ 1711]&	Returns the clipped starting point of the line.
aEndPoint	TcVnPoint2_REAL [▶ 1711]&	Returns the clipped end point of the line.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.3 ClipLineToBoundary (ITcVnImage*)

Clips a line to the boundary of an image.

Syntax

Definition:

```
HRESULT ClipLineToBoundary(
    HRESULT            hrPrev,
    TcVnVector4_LREAL& aLine,
    ITcVnImage*        ipImage,
    TcVnPoint2_REAL&   aStartPoint,
    TcVnPoint2_REAL&   aEndPoint
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aLine	TcVnVector4_LREAL [▶ 1713]&	The first and second element describe the x and y component of the direction vector. The third and fourth element describe the x and y component of the position vector.
ipImage	ITcVnImage* [▶ 1925]	Image, from which to derive the boundary
aStartPoint	TcVnPoint2_REAL [▶ 1711]&	Returns the clipped starting point of the line.
aEndPoint	TcVnPoint2_REAL [▶ 1711]&	Returns the clipped end point of the line.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.4 ClipLineToBoundary (TcVnRectangle_DINT)

Clips a line to a rectangular boundary.

Syntax

Definition:

```
HRESULT ClipLineToBoundary(
    HRESULT             hrPrev,
    TcVnVector4_LREAL&  aLine,
    TcVnRectangle_DINT& stRectangle,
    TcVnPoint2_REAL&    aStartPoint,
    TcVnPoint2_REAL&    aEndPoint
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aLine	TcVnVector4_LREAL [▶ 1713]&	The first and second element describe the x and y component of the direction vector. The third and fourth element describe the x and y component of the position vector.
stRectangle	TcVnRectangle_DINT [▶ 1781]&	Rectangular boundary
aStartPoint	TcVnPoint2_REAL [▶ 1711]&	Returns the clipped starting point of the line.
aEndPoint	TcVnPoint2_REAL [▶ 1711]&	Returns the clipped end point of the line.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.5 ConvertITcUnknownToITcVnBitmapExport

Convert an ITcUnknown interface pointer to an ITcVnBitmapExport interface pointer. The destination pointer will be released (if existing) and set to the source pointer. The source pointer will be set to 0.

Syntax

Definition:

```
HRESULT ConvertITcUnknownToITcVnBitmapExport(
    HRESULT             hrPrev,
    ITcUnknown*&        ipSrc,
    ITcVnBitmapExport*& ipDest
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrc	ITcUnknown* [▶ 1939]&	Source pointer
ipDest	ITcVnBitmapExport* [▶ 1911]&	Destination pointer

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.6 ConvertITcUnknownToITcVnContainer

Convert an ITcUnknown interface pointer to an ITcVnContainer interface pointer. The destination pointer will be released (if existing) and set to the source pointer. The source pointer will be set to 0.

Syntax

Definition:

```
HRESULT ConvertITcUnknownToITcVnContainer(
    HRESULT          hrPrev,
    ITcUnknown*&     ipSrc,
    ITcVnContainer*& ipDest
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrc	ITcUnknown* [▶ 1939]&	Source pointer
ipDest	ITcVnContainer* [▶ 1890]&	Destination pointer

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.7 ConvertITcUnknownToITcVnImage

Convert an ITcUnknown interface pointer to an ITcVnImage interface pointer. The destination pointer will be released (if existing) and set to the source pointer. The source pointer will be set to 0.

Syntax

Definition:

```
HRESULT ConvertITcUnknownToITcVnImage(
    HRESULT      hrPrev,
    ITcUnknown*& ipSrc,
    ITcVnImage*& ipDest
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrc	ITcUnknown* [▶ 1939]&	Source pointer
ipDest	ITcVnImage* [▶ 1925]&	Destination pointer

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.8 ConvertITcUnknownToITcVnMlModel

Convert an ITcUnknown interface pointer to an ITcVnMlModel interface pointer. The destination pointer will be released (if existing) and set to the source pointer. The source pointer will be set to 0.

Syntax

Definition:

```
HRESULT ConvertITcUnknownToITcVnMlModel(
    HRESULT        hrPrev,
    ITcUnknown*&   ipSrc,
    ITcVnMlModel*& ipDest
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrc	ITcUnknown* [▶ 1939]&	Source pointer
ipDest	ITcVnMlModel* [▶ 1933]&	Destination pointer

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.9 DeinitializeFunction

Deinitialize the specified options for the function.

Syntax

Definition:

```
HRESULT DeinitializeFunction(
    HRESULT                   hrPrev,
    ETcVnInitializableFunction eFunction,
    ULONGLONG                 nOptions
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
eFunction	ETcVnInitializableFunction [▶ 1743]	Initializable function
nOptions	ULONGLONG	Initialization options for the function

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.10 GetTimestamp

Get the timestamp (actual DC time) from the last change of an object (e.g. ITcVnImage or ITcVnContainer)

Syntax

Definition:

```
HRESULT GetTimestamp(
    HRESULT     hrPrev,
    ITcUnknown* ipUnknown,
    LONGLONG&   nTimestamp
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipUnknown	ITcUnknown* [▶ 1939]	An object implementing ITcUnknown and ITcVnTimestamp (e.g. ITcVnImage or ITcVnContainer)
nTimestamp	LONGLONG&	Returns the timestamp

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.11 HuMomentInvariants

Computes the Hu moment invariants.

Syntax

Definition:

```
HRESULT HuMomentInvariants(
    HRESULT           hrPrev,
    TcVnMoments&      stMoments,
    TcVnArray7_LREAL& aHuInvariants
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stMoments	TcVnMoments [▶ 1769]&	Moments from which to compute the Hu invariants
aHuInvariants	TcVnArray7_LREAL [▶ 1713]&	Returns the Hu moment invariants

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.12 InitMatrixStruct

Initialize a struct of the type TcVnMatrix extending a buffer with metainformation so that it can be used as a matrix.

Syntax

Definition:

```
HRESULT InitMatrixStruct(
    HRESULT          hrPrev,
    PVOID            pSrcBuffer,
    TcVnMatrix&      stDestMatrix,
    ULONG            nRows,
    ULONG            nCols,
    ETcVnElementType eElementType
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
pSrcBuffer	PVOID	Source buffer
stDestMatrix	TcVnMatrix [▶ 1769]&	Returns completed matrix struct
nRows	ULONG	Matrix rows
nCols	ULONG	Matrix columns
eElementType	ETcVnElementType [▶ 1738]	Type of the matrix elements

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.13 InvertMatrix3x3

Invert a 3x3 matrix.

Syntax

Definition:

```
HRESULT InvertMatrix3x3(
    HRESULT              hrPrev,
    TcVnMatrix3x3_LREAL& aSrcMatrix,
    TcVnMatrix3x3_LREAL& aInvertedMatrix
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aSrcMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	Source matrix
aInvertedMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	Returns inverted matrix

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.14 LineIntersectionPoint

Computes the intersection point between two lines (returns S_FALSE if the provided lines are parallel).

Syntax

Definition:

```
HRESULT LineIntersectionPoint(
    HRESULT            hrPrev,
    TcVnVector4_LREAL& aLine1,
    TcVnVector4_LREAL& aLine2,
    TcVnPoint2_REAL&   aIntersecPoint
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aLine1	TcVnVector4_LREAL [▶ 1713]&	First line. The first and second element describe the x and y component of a vector collinear to the line. The third and fourth element describe the x and y component of a point on the line.
aLine2	TcVnVector4_LREAL [▶ 1713]&	Second line. The first and second element describe the x and y component of a vector collinear to the line. The third and fourth element describe the x and y component of a point on the line.
aIntersecPoint	TcVnPoint2_REAL [▶ 1711]&	Returns the intersection point of aLine1 and aLine2.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.15 LineIntersectionPointAndAngle

Computes the intersection point and angle between two lines (returns S_FALSE if the provided lines are parallel).

Syntax

Definition:

```
HRESULT LineIntersectionPointAndAngle(
    HRESULT            hrPrev,
    TcVnVector4_LREAL& aLine1,
    TcVnVector4_LREAL& aLine2,
    TcVnPoint2_REAL&   aIntersecPoint,
    float&             fAngle,
    bool               bAngleInDegrees
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aLine1	TcVnVector4_LREAL [▶ 1713]&	First line. The first and second element describe the x and y component of a vector collinear to the line. The third and fourth element describe the x and y component of a point on the line.
aLine2	TcVnVector4_LREAL [▶ 1713]&	Second line. The first and second element describe the x and y component of a vector collinear to the line. The third and fourth element describe the x and y component of a point on the line.
aIntersecPoint	TcVnPoint2_REAL [▶ 1711]&	Returns the intersection point of aLine1 and aLine2.
fAngle	float&	Returns the intersection angle of aLine1 and aLine2.
bAngleInDegrees	bool	If TRUE, fAngle is in degrees. If FALSE, fAngle is in radians.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.16 MultiplyMatrices

Matrix multiplication of two matrices (A * B = C). All matrices are represented by structs holding a pointer to an array of the appropriate size.

Syntax

Definition:

```
HRESULT MultiplyMatrices(
    HRESULT     hrPrev,
    TcVnMatrix& stSrcMatrix1,
    TcVnMatrix& stSrcMatrix2,
    TcVnMatrix& stDestMatrix
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stSrcMatrix1	TcVnMatrix [▶ 1769]&	First source matrix (A)
stSrcMatrix2	TcVnMatrix [▶ 1769]&	Second source matrix (B)
stDestMatrix	TcVnMatrix [▶ 1769]&	Destination matrix (C) (The destination matrix is filled by this function, but the required memory needs to be provided.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.17 RotatedRectangleCorners

Computes the 4 corner points of a rotated rectangle.

Syntax

Definition:

```
HRESULT RotatedRectangleCorners(
    HRESULT                 hrPrev,
    TcVnRotatedRectangle&   stRect,
    TcVnArray4_Point2_REAL& aCorners
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stRect	TcVnRotatedRectangle [▶ 1782]&	Rotated rectangle
aCorners	TcVnArray4_Point2_REAL [▶ 1713]&	Returns the 4 corner points

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.18 RotatedRectangleIntersection

Finds the intersection of 2 rotated rectangles.

Syntax

Definition:

```
HRESULT RotatedRectangleIntersection(
    HRESULT                     hrPrev,
    TcVnRotatedRectangle&       stRect1,
    TcVnRotatedRectangle&       stRect2,
    ITcVnContainer*&            ipIntersection,
    ETcVnRectangleIntersection& eIntersection
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
stRect1	TcVnRotatedRectangle [▶ 1782]&	Rotated rectangle 1
stRect2	TcVnRotatedRectangle [▶ 1782]&	Rotated rectangle 2
ipIntersection	ITcVnContainer* [▶ 1890]&	Returns the intersection points (ContainerType_Vector_TcVnPoint2_REAL, non-zero interface pointers are reused)
eIntersection	ETcVnRectangleIntersection [▶ 1752]&	Returns the intersection type

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.19 SetRngSeed

Sets the internal pseudo random number generator seed (intended for testing purposes only).

Syntax

Definition:

```
HRESULT SetRngSeed(
    HRESULT hrPrev,
    LONG    nSeed
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
nSeed	LONG	seed (0 sets the generator back to its initial state)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.20 UpdateTimestamp

Update the timestamp of an object to the actual DC time (e.g. ITcVnImage or ITcVnContainer)

Syntax

Definition:

```
HRESULT UpdateTimestamp(
    HRESULT     hrPrev,
    ITcUnknown* ipUnknown
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipUnknown	ITcUnknown* [▶ 1939]	An object implementing ITcUnknown and ITcVnTimestamp (e.g. ITcVnImage or ITcVnContainer)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.21 OrthogonalVector (TcVnVector2_DINT)

Computes an orthogonal vector to a given source vector.

Syntax

Definition:

```
HRESULT OrthogonalVector(
    HRESULT            hrPrev,
    TcVnVector2_DINT&  aSrc,
    TcVnVector2_LREAL& aOrthogonal,
    bool               bNormalize
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aSrc	TcVnVector2_DINT [▶ 1713]&	Source vector
aOrthogonal	TcVnVector2_LREAL [▶ 1713]&	Returns a vector orthogonal to aSrc
bNormalize	bool	If true, aOrthogonal length is normalized to 1, otherwise aOrthogonal has the same length as aSrc

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.22 OrthogonalVector (TcVnVector2_LREAL)

Computes an orthogonal vector to a given source vector.

Syntax

Definition:

```
HRESULT OrthogonalVector(
    HRESULT            hrPrev,
    TcVnVector2_LREAL& aSrc,
```

```
    TcVnVector2_LREAL& aOrthogonal,
    bool               bNormalize
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aSrc	TcVnVector2_LREAL [▶ 1713]&	Source vector
aOrthogonal	TcVnVector2_LREAL [▶ 1713]&	Returns a vector orthogonal to aSrc
bNormalize	bool	If true, aOrthogonal length is normalized to 1, otherwise aOrthogonal has the same length as aSrc

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.23 OrthogonalVector (TcVnVector2_REAL)

Computes an orthogonal vector to a given source vector.

Syntax

Definition:

```
HRESULT OrthogonalVector(
    HRESULT           hrPrev,
    TcVnVector2_REAL& aSrc,
    TcVnVector2_REAL& aOrthogonal,
    bool              bNormalize
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aSrc	TcVnVector2_REAL [▶ 1713]&	Source vector
aOrthogonal	TcVnVector2_REAL [▶ 1713]&	Returns a vector orthogonal to aSrc
bNormalize	bool	If true, aOrthogonal length is normalized to 1, otherwise aOrthogonal has the same length as aSrc

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.24 PointToLineDistance (TcVnPoint2_DINT)

Computes the shortest distance between a point and a line.

Syntax

Definition:

```
HRESULT PointToLineDistance(
    HRESULT          hrPrev,
    TcVnPoint2_DINT& aPoint,
    TcVnPoint2_DINT& aLinePoint1,
    TcVnPoint2_DINT& aLinePoint2,
    double&          fDistance
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aPoint	TcVnPoint2_DINT [▶ 1711]&	Point
aLinePoint1	TcVnPoint2_DINT [▶ 1711]&	Line Point 1
aLinePoint2	TcVnPoint2_DINT [▶ 1711]&	Line Point 2
fDistance	double&	Returns the distance between the point and the line

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.25 PointToLineDistance (TcVnPoint2_LREAL)

Computes the shortest distance between a point and a line.

Syntax

Definition:

```
HRESULT PointToLineDistance(
    HRESULT           hrPrev,
    TcVnPoint2_LREAL& aPoint,
    TcVnPoint2_LREAL& aLinePoint1,
    TcVnPoint2_LREAL& aLinePoint2,
    double&           fDistance
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aPoint	TcVnPoint2_LREAL [▶ 1711]&	Point
aLinePoint1	TcVnPoint2_LREAL [▶ 1711]&	Line Point 1
aLinePoint2	TcVnPoint2_LREAL [▶ 1711]&	Line Point 2
fDistance	double&	Returns the distance between the point and the line

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.26 PointToLineDistance (TcVnPoint2_REAL)

Computes the shortest distance between a point and a line.

Syntax

Definition:

```
HRESULT PointToLineDistance(
    HRESULT          hrPrev,
    TcVnPoint2_REAL& aPoint,
    TcVnPoint2_REAL& aLinePoint1,
    TcVnPoint2_REAL& aLinePoint2,
    double&          fDistance
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aPoint	TcVnPoint2_REAL [▶ 1711]&	Point
aLinePoint1	TcVnPoint2_REAL [▶ 1711]&	Line Point 1
aLinePoint2	TcVnPoint2_REAL [▶ 1711]&	Line Point 2
fDistance	double&	Returns the distance between the point and the line

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.27 PointToPointDirection (TcVnPoint2_DINT)

Computes the direction vector between 2 points.

Syntax

Definition:

```
HRESULT PointToPointDirection(
    HRESULT            hrPrev,
    TcVnPoint2_DINT&   aPoint1,
    TcVnPoint2_DINT&   aPoint2,
    TcVnVector2_LREAL& aDirection,
    bool               bNormalize
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aPoint1	TcVnPoint2_DINT [▶ 1711]&	Point 1
aPoint2	TcVnPoint2_DINT [▶ 1711]&	Point 2
aDirection	TcVnVector2_LREAL [▶ 1713]&	Returns the direction vector from aPoint1 to aPoint2
bNormalize	bool	If true, aDirection length is normalized to 1, otherwise aDirection length is the distance between aPoint1 and aPoint2

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.28 PointToPointDirection (TcVnPoint2_LREAL)

Computes the direction vector between 2 points.

Syntax

Definition:

```
HRESULT PointToPointDirection(
    HRESULT            hrPrev,
    TcVnPoint2_LREAL&  aPoint1,
    TcVnPoint2_LREAL&  aPoint2,
    TcVnVector2_LREAL& aDirection,
    bool               bNormalize
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aPoint1	TcVnPoint2_LREAL [▶ 1711]&	Point 1
aPoint2	TcVnPoint2_LREAL [▶ 1711]&	Point 2
aDirection	TcVnVector2_LREAL [▶ 1713]&	Returns the direction vector from aPoint1 to aPoint2
bNormalize	bool	If true, aDirection length is normalized to 1, otherwise aDirection length is the distance between aPoint1 and aPoint2

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.29 PointToPointDirection (TcVnPoint2_REAL)

Computes the direction vector between 2 points.

Syntax

Definition:

```
HRESULT PointToPointDirection(
    HRESULT           hrPrev,
    TcVnPoint2_REAL&  aPoint1,
    TcVnPoint2_REAL&  aPoint2,
    TcVnVector2_REAL& aDirection,
    bool              bNormalize
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aPoint1	TcVnPoint2_REAL [▶ 1711]&	Point 1
aPoint2	TcVnPoint2_REAL [▶ 1711]&	Point 2
aDirection	TcVnVector2_REAL [▶ 1713]&	Returns the direction vector from aPoint1 to aPoint2
bNormalize	bool	If true, aDirection length is normalized to 1, otherwise aDirection length is the distance between aPoint1 and aPoint2

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.30 PointToPointDistance (TcVnPoint2_DINT)

Computes the distance between 2 points.

Syntax

Definition:

```
HRESULT PointToPointDistance(
    HRESULT          hrPrev,
    TcVnPoint2_DINT& aPoint1,
    TcVnPoint2_DINT& aPoint2,
    double&          fDistance
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aPoint1	TcVnPoint2_DINT [▶ 1711]&	Point 1
aPoint2	TcVnPoint2_DINT [▶ 1711]&	Point 2
fDistance	double&	Returns the distance between the points

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.31 PointToPointDistance (TcVnPoint2_LREAL)

Computes the distance between 2 points.

Syntax

Definition:

```
HRESULT PointToPointDistance(
    HRESULT           hrPrev,
    TcVnPoint2_LREAL& aPoint1,
    TcVnPoint2_LREAL& aPoint2,
    double&           fDistance
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aPoint1	TcVnPoint2_LREAL [▶ 1711]&	Point 1
aPoint2	TcVnPoint2_LREAL [▶ 1711]&	Point 2
fDistance	double&	Returns the distance between the points

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.21.32 PointToPointDistance (TcVnPoint2_REAL)

Computes the distance between 2 points.

Syntax

Definition:

```
HRESULT PointToPointDistance(
    HRESULT          hrPrev,
    TcVnPoint2_REAL& aPoint1,
    TcVnPoint2_REAL& aPoint2,
    double&          fDistance
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
aPoint1	TcVnPoint2_REAL [▶ 1711]&	Point 1
aPoint2	TcVnPoint2_REAL [▶ 1711]&	Point 2
fDistance	double&	Returns the distance between the points

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.22 Neural Network

Diese Gruppe enthält Funktionen zum Pre- und Postprocessing, sowie zur Ausführung von Neural Network Modellen.

Funktionen

Pre- & Postprocessing

- ConvertDataLayout [▶ 2709]
- ReshapeImage [▶ 2723]

Ausführung von Neuronalen Netzen

- ExecuteNeuralNetwork [▶ 2710]
  Mit einem Eingang und einem Ausgang.
- ExecuteNeuralNetwork_MI [▶ 2711]
  Mit mehreren Eingängen (MI) und einem Ausgang.
- ExecuteNeuralNetwork_MIMO_String [▶ 2712]
  Mit mehreren Eingängen und Ausgängen (MIMO).
- ExecuteNeuralNetwork_MISO [▶ 2713]
  Mit mehreren Eingängen und einem einzigen spezifizierten Ausgang (MISO).

- ExecuteNeuralNetwork_MO [▶ 2714]
  Ein Eingang und mehrere Ausgänge (MO).
- ExecuteNeuralNetwork_SO [▶ 2715]
  Mit einem Eingang und einem einzigen spezifizierten Ausgang (SO).

Postprocessing

- Sigmoid [▶ 2724]
- SoftMax [▶ 2725]
- NonMaxSuppression [▶ 2720]
- Reduce [▶ 2722]
- ReduceArg [▶ 2723]

Abfrage der Layer-Bezeichnungen

- GetNeuralNetworkLayerNames [▶ 2718]

6.2.3.22.1 ConvertDataLayout

Convert a data layout (the arrangement of dimensions in the input data) to another data layout.

Syntax

Definition:

```
HRESULT ConvertDataLayout(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage,
    ULONGLONG    nSrcDataLayout,
    ULONGLONG    nDestDataLayout
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image will be created if required.)
nSrcDataLayout	ULONGLONG	Data layout of the source image (Commonly used data layouts are available in the enumeration ETcVnDataLayout. If nSrcDataLayout is set to 0 or TCVN_DL_DEFAULT, the DataLayout will be automatically determined based on the dimension of the ipSrcImage: 0x12 for 2D, 0x123 for 3D, 0x1234 for 4D, and so forth.)
nDestDataLayout	ULONGLONG	Required output data layout (Commonly used data layouts are available in the enumeration ETcVnDataLayout. nDestDataLayout must not equal 0 or TCVN_DL_DEFAULT.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.22.2 ExecuteNeuralNetwork

Execute a neural network using the provided image as input and provide the output of one output layer. The function doesn't perform any pre-processing or transformation.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT ExecuteNeuralNetwork(
    HRESULT             hrPrev,
    ITcVnNeuralNetwork* ipNeuralNetwork,
    ITcVnImage*         ipSrcImage,
    ITcVnImage*&        ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipNeuralNetwork	ITcVnNeuralNetwork* [▶ 1941]	Neural network model
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (The image layout must be adapted to match the model's expected input format using e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage functions if necessary.)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image based on the output layer is created. To process the destination image, use e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage if necessary.

Return value

HRESULT [▶ 150]

Required License

TC3 Neural Network Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.22.3 ExecuteNeuralNetwork (MI)

Execute a neural network with multiple inputs (MI) using the provided images. Ensure all model input names are listed in sInputNames and that the images and names are in the same order. The result of one output layer is returned. The function doesn't perform any pre-processing or transformation.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT ExecuteNeuralNetwork(
    HRESULT             hrPrev,
    ITcVnNeuralNetwork* ipNeuralNetwork,
    PVOID               pSrcImages,
    USHORT              nSrcArraySize,
    PCCH                sInputNames,
    ITcVnImage*&        ipDestImage
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipNeuralNetwork	ITcVnNeuralNetwork* [▶ 1941]	Neural network model
pSrcImages	PVOID	Pointer to an array of source input images
nSrcArraySize	USHORT	The size of the array referenced by pSrcImages. pSrcImages must reference an array with a size equal to the number of model inputs.
sInputNames	PCCH	Input names separated by ";"
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image based on the output layer is created. To process the destination image, use e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage if necessary.

Return value

HRESULT [▶ 150]

Required License

TC3 Neural Network Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.22.4 ExecuteNeuralNetwork (MIMO_String)

Execute a neural network with multiple inputs and multiple outputs (MIMO) using the provided images. Ensure all model input names are listed in sInputNames and that the images and names are in the same order. The result of the specified output layers are returned. The function doesn't perform any pre-processing or transformation.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT ExecuteNeuralNetwork(
    HRESULT             hrPrev,
    ITcVnNeuralNetwork* ipNeuralNetwork,
    PVOID               pSrcImages,
    USHORT              nSrcArraySize,
    PCCH                sInputNames,
    ITcVnContainer*&    ipDestImages,
    PCCH                sOutputNames
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipNeuralNetwork	ITcVnNeuralNetwork* [▶ 1941]	Neural network model
pSrcImages	PVOID	Pointer to an array of source input images
nSrcArraySize	USHORT	The size of the array referenced by pSrcImages. pSrcImages must reference an array with a size equal to the number of model inputs.
sInputNames	PCCH	Input names separated by ";"
ipDestImages	ITcVnContainer* [▶ 1890]&	Container of images containing the output from the specified layers.
sOutputNames	PCCH	Names of output layers whose outputs are required separated by ";". If empty, the outputs of all unconnected output layers are provided.

Return value

HRESULT [▶ 150]

Required License

TC3 Neural Network Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.22.5 ExecuteNeuralNetwork (MISO)

Execute a neural network with multiple inputs and single output (MISO) using the provided images. Ensure all model input names are listed in sInputNames and that the images and names are in the same order. The result of the specified output layer is returned. The function doesn't perform any pre-processing or transformation.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT ExecuteNeuralNetwork(
    HRESULT             hrPrev,
    ITcVnNeuralNetwork* ipNeuralNetwork,
    PVOID               pSrcImages,
    USHORT              nSrcArraySize,
    PCCH                sInputNames,
    ITcVnImage*&        ipDestImage,
    PCCH                sOutputName
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipNeuralNetwork	ITcVnNeuralNetwork* [▶ 1941]	Neural network model
pSrcImages	PVOID	Pointer to an array of source input images
nSrcArraySize	USHORT	The size of the array referenced by pSrcImages. pSrcImages must reference an array with a size equal to the number of model inputs.
sInputNames	PCCH	Input names separated by ";"
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image based on the output layer is created. To process the destination image, use e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage if necessary.
sOutputName	PCCH	Specify the name of the layer for which the output is required. If empty, the output of one output layer is provided.

Return value

HRESULT [▶ 150]

Required License

TC3 Neural Network Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.22.6 ExecuteNeuralNetwork (MO)

Execute a neural network using the provided image as input and provide the output of the specified output layers. The function doesn't perform any pre-processing or transformation.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT ExecuteNeuralNetwork(
    HRESULT             hrPrev,
    ITcVnNeuralNetwork* ipNeuralNetwork,
    ITcVnImage*         ipSrcImage,
    ITcVnContainer*&    ipDestImages,
    ITcVnContainer*     ipOutputNames
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipNeuralNetwork	ITcVnNeuralNetwork* [▶ 1941]	Neural network model
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (The image layout must be adapted to match the model's expected input format using e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage functions if necessary.)
ipDestImages	ITcVnContainer* [▶ 1890]&	Container of images containing the output from the specified layers.
ipOutputNames	ITcVnContainer* [▶ 1890]	Container of layer names whose outputs to be returned (ContainerType_Vector_String_SINT). If set to zero, the outputs of all unconnected output layers will be returned..

Return value

HRESULT [▶ 150]

Required License

TC3 Neural Network Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.22.7 ExecuteNeuralNetwork (SO)

Execute a neural network using the provided image as input and provide the output of the specified output layer. The function doesn't perform any pre-processing or transformation.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT ExecuteNeuralNetwork(
    HRESULT             hrPrev,
    ITcVnNeuralNetwork* ipNeuralNetwork,
    ITcVnImage*         ipSrcImage,
    ITcVnImage*&        ipDestImage,
    PCCH                sOutputName
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipNeuralNetwork	ITcVnNeuralNetwork* [▶ 1941]	Neural network model
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (The image layout must be adapted to match the model's expected input format using e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage functions if necessary.)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image based on the output layer is created. To process the destination image, use e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage if necessary.
sOutputName	PCCH	Specify the name of the layer for which the output is required. If empty, the output of one output layer is provided.

Return value

HRESULT [▶ 150]

Required License

TC3 Neural Network Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.22.8 ExecuteNeuralNetwork_MultiLayerOutput

This function is deprecated, please use F_VN_ExecuteNeuralNetwork_MO. Execute a neural network using the provided image as input and provide the output of the specified output layers. The function doesn't perform any pre-processing or transformation.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT ExecuteNeuralNetwork_MultiLayerOutput(
    HRESULT             hrPrev,
    ITcVnNeuralNetwork* ipNeuralNetwork,
    ITcVnImage*         ipSrcImage,
    ITcVnContainer*&    ipDestImages,
    ITcVnContainer*     ipLayerNames
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipNeuralNetwork	ITcVnNeuralNetwork* [▶ 1941]	Neural network model
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (The image layout must be adapted to match the model's expected input format using e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage functions if necessary.)
ipDestImages	ITcVnContainer* [▶ 1890]&	Container of images containing the output from the specified layers.
ipLayerNames	ITcVnContainer* [▶ 1890]	Container of layer names whose outputs to be returned (ContainerType_Vector_String_SINT). If set to zero, the outputs of all unconnected output layers will be returned..

Return value

HRESULT [▶ 150]

Required License

TC3 Neural Network Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.22.9 ExecuteNeuralNetwork_SingleLayerOutput

This function is deprecated, please use F_VN_ExecuteNeuralNetwork_SO. Execute a neural network using the provided image as input and provide the output of the specified output layer. The function doesn't perform any pre-processing or transformation.
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT ExecuteNeuralNetwork_SingleLayerOutput(
    HRESULT             hrPrev,
    ITcVnNeuralNetwork* ipNeuralNetwork,
    ITcVnImage*         ipSrcImage,
    ITcVnImage*&        ipDestImage,
    PCCH                sLayerName
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipNeuralNetwork	ITcVnNeuralNetwork* [▶ 1941]	Neural network model
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (The image layout must be adapted to match the model's expected input format using e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage functions if necessary.)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate destination image based on the output layer is created. To process the destination image, use e.g. F_VN_ConvertDataLayout or F_VN_ReshapeImage if necessary.
sLayerName	PCCH	Specify the name of the layer for which the output is required. If empty, the output of one output layer is provided.

Return value

HRESULT [▶ 150]

Required License

TC3 Neural Network Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.22.10 GetNeuralNetworkLayerNames

Get the name of all layers in the neural network model. If the bOnlyOuput flag is set to true, only the unconnected output layer names are returned.

Syntax

Definition:

```
HRESULT GetNeuralNetworkLayerNames(
    HRESULT             hrPrev,
    ITcVnNeuralNetwork* ipNeuralNetwork,
    ITcVnContainer*&    ipLayerNames,
    bool                bOnlyOutputs
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipNeuralNetwork	ITcVnNeuralNetwork* [▶ 1941]	Neural network model
ipLayerNames	ITcVnContainer* [▶ 1890]&	Container with the layers names (ContainerType_Vector_String_SINT)
bOnlyOutputs	bool	If true, only the unconnected output layer names are returned.

Return value

HRESULT [▶ 150]

Required License

TC3 Neural Network Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.22.11 GetNeuralNetworkLayerNames (Exp)

Get the name of model layers in the neural network model.

Syntax

Definition:

```
HRESULT GetNeuralNetworkLayerNames(
    HRESULT             hrPrev,
    ITcVnNeuralNetwork* ipNeuralNetwork,
```

```
    ITcVnContainer*&    ipLayerNames,
    ETcVnLayerNameType  eLayerNameType
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipNeuralNetwork	ITcVnNeuralNetwork* [▶ 1941]	Neural network model
ipLayerNames	ITcVnContainer* [▶ 1890]&	Container with the layers names (ContainerType_Vector_String_SINT)
eLayerNameType	ETcVnLayerNameType [▶ 1762]	Layer name type

Return value

HRESULT [▶ 150]

Required License

TC3 Neural Network Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.22.12 GetNeuralNetworkLayerNames (String)

Get the name of model layers in the neural network model.

Syntax

Definition:

```
HRESULT GetNeuralNetworkLayerNames(
    HRESULT             hrPrev,
    ITcVnNeuralNetwork* ipNeuralNetwork,
    PCCH                sLayerNames,
    ULONGLONG           nMaxLength,
    ETcVnLayerNameType  eLayerNameType
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipNeuralNetwork	ITcVnNeuralNetwork* [▶ 1941]	Neural network model
sLayerNames	PCCH	Layer names separated by ";". Make sure to choose a sufficient STRING size!
nMaxLength	ULONGLONG	Maximum string length to export (including 0 termination). If the container content is longer, the string is cut off at nMaxLength - 1 and 0 termination is appended. In this case, S_FALSE is returned.
eLayerNameType	ETcVnLayerNameType [▶ 1762]	Layer name type

Return value

HRESULT [▶ 150]

Required License

TC3 Neural Network Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.22.13 NonMaxSuppression

Perform non-maximum suppression on given bounding boxes and corresponding scores.

Syntax

Definition:

```
HRESULT NonMaxSuppression(
    HRESULT          hrPrev,
    ITcVnContainer*  ipBoundingBoxes,
    ITcVnContainer*  ipScores,
    ITcVnContainer*& ipDestIndcies,
    float            fScoreThreshold,
    float            fOverlapThreshold,
    ITcVnContainer*  ipClassIndcies = nullptr,
    float            fEta = 1.0f,
    LONG             nTopK = 0
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipBoundingBoxes	ITcVnContainer* [▶ 1890]		Container of bounding boxes to apply the non maximum supresion (ContainerType_Vector_TcVnRectangle_DINT)
ipScores	ITcVnContainer* [▶ 1890]		Container of coreeponding score for each bounding box (ContainerType_Vector_REAL)
ipDestIndcies	ITcVnContainer* [▶ 1890]&		Container of the kept indcies after apply the non maximum suppression (ContainerType_Vector_DINT)
fScoreThreshold	float		A threshold [0.0 to 1.0] used to filter the bounding boxes by score
fOverlapThreshold	float		A threshold [0.0 to 1.0] that determines the minimum degree of intersection over union (IoU) required for bounding boxes to be considered overlapping
ipClassIndcies	ITcVnContainer* [▶ 1890]	nullptr	Container of corresponding set of class indcies (ContainerType_Vector_DINT)
fEta	float	1.0f	An adjustable threshold [0.0 to 1.0] used to dynamically reduce the overlap threshold during processing. Setting it to 1 maintains a constant value for the overlap threshold
nTopK	LONG	0	The maximum number of bounding boxes with the highest scores to undergo the suppression algorithm (must be >= 0). Setting it to zero applies the algorithm to all bounding boxes

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.22.14 Reduce

Reduce ipSrcImage by applying the specified operation across rows/columns to obtain a single row/column/channel with the resulting values.

Syntax

Definition:

```
HRESULT Reduce(
    HRESULT         hrPrev,
    ITcVnImage*     ipSrcImage,
    ITcVnImage*&    ipDestImage,
    LONG            nDim,
    ETcVnReduceType eReduceType
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (USINT or REAL elements, 1 channel)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image. If ipSrcImage is of type USINT and the reduction type is SUM or AVG, ipDestImage will be of type REAL; otherwise, it retains the same type as ipSrcImage (An appropriate single channel destination image will be created if required.)
nDim	LONG	Index along which the ITcVnImage is reduced: 0 reduces the matrix to a single row, 1 reduces it to a single column, and 2 reduces to a single channel
eReduceType	ETcVnReduceType [▶ 1761]	Reduction operation

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.22.15 ReduceArg

Finds indices of max/min elements along provided axis and reduce the ipSrcImage to a single row/column/channel.

Syntax

Definition:

```
HRESULT ReduceArg(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage,
    LONG         nDim,
    LONG         eReduceType
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image of type DINT (An appropriate single channel destination image will be created if required.)
nDim	LONG	Index along which the ipSrcImage is reduced: 0 reduces the matrix to a single row, 1 reduces it to a single column, and 2 reduces to a single channel
eReduceType	LONG	Reduction operation REDUCE_MAX or REDUCE_MIN

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.22.16 ReshapeImage

Changes the shape of an ITcVnImage. The total number of elements in the new shape must be equal to the total number of elements in ipSrcImage.

Syntax

Definition:

```
HRESULT ReshapeImage(
    HRESULT            hrPrev,
    ITcVnImage*        ipSrcImage,
    ITcVnImage*&       ipDestImage,
    TcVnArray33_UDINT& aNewShape
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (1 channel)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate single channel destination image will be created if required.)
aNewShape	TcVnArray33_UDINT [▶ 1713]&	Specifies the shape of ipDestImage. First entry is the number of dimensions (max. 32), followed by the sizes of each dimension. If one dimension is set to zero, its value will be automatically calculated based on the total elements of ipSrcImage (Only one dimension can be set to zero.)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.22.17 Sigmoid

Apply the logistic function 1 / (1 + exp(-fSteepness * (x - fShift))) element-wise to a single- or multi-channel REAL image.

Syntax

Definition:

```
HRESULT Sigmoid(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage,
    float        fShift = 0.0f,
    float        fSteepness = 1.0f
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (REAL)
ipDestImage	ITcVnImage* [▶ 1925]&		Destination image (An appropriate destination image will be created if required.)
fShift	float	0.0f	Determines the intensity level around which the logistic function transitions.
fSteepness	float	1.0f	Controls the sharpness of the transition.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.22.18 SoftMax

Apply softmax function pixel-wise to a single channel REAL image.

Syntax

Definition:

```
HRESULT SoftMax(
    HRESULT      hrPrev,
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipDestImage,
    bool         bFlatten
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (REAL, 1 channel)
ipDestImage	ITcVnImage* [▶ 1925]&	Destination image (An appropriate single channel destination image will be created if required.)
bFlatten	bool	If true, the resulting image will be reshaped into a single row.

Return value

HRESULT [▶ 150]

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.23 OCR

Diese Gruppe enthält Funktionen zur optischen Zeichenerkennung (**O**ptical **C**haracter **R**ecognition).

Die OCR-Funktionen [▶ 2728] identifizieren Zeichen in einem Bild und geben die erkannte Zeichenkette als String zurück. Die Klassifizierung basiert auf klassischen Machine Learning Modellen. Die Modelle werden trainiert bereitgestellt, so dass die Funktionen direkt angewendet werden können. Zusätzliche Einstellungen oder ein eigenes Training sind nicht erforderlich. Die unterstützen Zeichen und Fonts hängen von den im Training verwendeten Daten ab. Daher gibt es verschiedene Modelle deren Funktionsumfang, sowie die generellen Vorbedingungen und Anforderungen, im Folgenden beschrieben werden.

Für alle Zeichen, die in einem Bild erkannt werden, ordnen die Modelle die einzelnen Zeichen einer bekannten Klasse zu, bei der die Übereinstimmung am höchsten ist. Das Ergebnis besteht also immer aus den bekannten Zeichen des verwendeten Modelles. Eine Zurückweisung oder Aussortierung unbekannter Zeichen ist nicht enthalten.

Die Funktionen unterstützen nur einzeilige Zeichenfolgen, mehrere Zeilen müssen in einzelne ROIs unterteilt und mit mehreren Funktionsaufrufen gelesen werden.

Generelle Anforderungen zur Erkennung von Zeichen:

- Zeichenhöhe min. 20 Pixel
- Strichstärke min. 3 Pixel
- Punkte min. 3 x 3 Pixel
- Striche min. 3 x 6 Pixel
- Zeichenabstand / -trennung min. 4 Pixel
- Zeichen dürfen sich nicht überlappen
- Nur horizontale Ausrichtung / Anordnung der Zeichen max. ± 6° Abweichung
- Der Strich / Kontur eines Zeichens darf nicht unterbrochen sein
- Die Zeichen dürfen nicht gespiegelt oder auf den Kopf gestellt sein

Generelle Anforderungen an das Bild:

- ROI nur mit den zu erkennenden Zeichen und einer störungsfreien Zone um die Zeichen
- Innerhalb der ROI dürfen die Zeichen nicht von einer weiteren Kontur wie z.B. einem Rechteck umschlossen sein
- Guter Kontrast zwischen Zeichen und Hintergrund
- Homogener, nicht verrauschter oder gestörter, intransparenter Hintergrund

Anforderungen an die Fonts:

- Nur proportionale Schriftarten mit gleichen Zeichenbreiten
- Größere Lücken werden immer nur als ein Leerzeichen erkannt
- Nur serifenlose Schriftarten wie z.B. Arial, Tahoma, Courier, Univers, Frutiger, Verdana, OCR-B
- Keine gemischten Schriftarten
- Keine Punktdruck (dot print) oder kursiv (italic) Fonts

Modelle:

Das Enum ETcVnOcrModelType [▶ 1748] ermöglicht den Zugriff auf die folgenden Modelle:

- `OMT_NUMBERS`
  - ermöglich die Klassifizierung von Zahlen
  - enthält die Zeichen von `0-9`
- `OMT_NUMBERS_SC`
  - ermöglich die Klassifizierung von Zahlen und Sonderzeichen
  - enthält die Zeichen von `0-9` und sechs Sonderzeichen `. / - : = +`
- `OMT_UCLETTERS`
  - ermöglich die Klassifizierung von Großbuchstaben
  - enthält die Zeichen von `A-Z`
- `OMT_NUMBERS_SC_UCLETTERS`
  - ermöglich die Klassifizierung von Zahlen, Sonderzeichen und Großbuchstaben
  - enthält die Zeichen von `0-9`, die sechs Sonderzeichen `. / - : =` + und `A-Z`

Bei der Verwendung des kombinierten Modells (`OMT_NUMBERS_SC_UCLETTERS`) kann es, aufgrund der großen Ähnlichkeit von bestimmten Zeichen, zu Verwechselungen kommen. Beispiele dafür sind O und 0, S und 5 sowie B und 8.

Wenn bekannt ist an welcher Stelle eine Zahl oder ein Buchstabe steht, sollte alternativ die optionale Formatierungsvorgabe (`sPattern`) und der Kombination von den separaten Modellen verwendet werden. Alternativ können im Anschluss die Zeichen des Ergebnisstrings (`ipCharacters`) auch einzeln analysiert werden, so dass man individuell entscheiden kann ob z.B. anstatt O auch eine 0 akzeptiert wird.

Funktionen initialisieren

Um die OCR-Funktion mit einem oder mehreren Modellen verwenden zu können, muss die Funktion zuvor über den Baustein InitializeFunction [▶ 2747] mit den jeweiligen Modellen initialisiert werden.

Die Dauer für das Laden der Modelle ist abhängig von der Leistungsfähigkeit des verwendeten IPC, sowie der Modellgröße und kann mehrere hundert Millisekunden, bis hin zu mehreren Sekunden dauern. Daher ist bei kürzeren Taskzykluszeiten mit Zykluszeitüberschreitungen zu rechnen.

Wenn das kombinierte Modell `OMT_NUMBERS_SC_UCLETTERS` verwendet werden soll, muss aufgrund der Dateigröße der Router Speicher auf mindestens 512 MB gesetzt werden.

Die genaue Größe ist abhängig von der sonstigen Verwendung des Router Speichers und evtl. vorhandener Fragmentierung. Wenn der Baustein `InitializeFunction` z.B. einen Return Code `0x80004005` zurückgibt, deutet das auf zu wenig Router Speicher hin. Daher sollte zuerst der Router Speicher geprüft und ggf. vergrößert werden. Nach dem Laden eines Modells wird ein Großteil des Speichers wieder freigegeben und steht für die Anwendung zur Verfügung.

Interpretation des HRESULT

- Bei der Standardfunktion wird `S_OK` zurückgegeben, wenn Zeichen auf dem Bild erkannt wurden. Bei der Verwendung von optionalen Parametern hängt es davon ab, ob ein `sPattern` angegeben wurde. Wenn keine Formatierungsvorgabe angegeben wurde, entspricht die Rückgabe der Standardfunktion. Wenn eine Formatierungsvorgabe übergeben wurde, wird nur `S_OK` zurückgegeben, wenn die erkannten Zeichen mit der Pattern-Vorgabe übereinstimmen.
- Wenn die Standardfunktion erfolgreich ausgeführt wurde und keine Zeichen gefunden wurden oder bei optionaler Pattern-Vorgabe die gefundenen Zeichen damit nicht übereinstimmen, wird `S_FALSE` zurückgegeben.
- Wenn zu einer optionalen `sPattern` Angaben das übergebene Modell fehlt, wird ein `HRESULT = ADSERR_DEVICE_NOTINIT` zurückgegeben.
- Wenn der Standardfunktion mehrere Modelle übergeben werden, wird immer ein `HRESULT = ADSERR_DEVICE_INVALIDPARM` zurückgegeben, da die Funktion pro Aufruf nur ein Modell unterstützt. Bei der Verwendung von optionalen Parametern hängt es davon ab, ob ein `sPattern` angegeben wurde. Wenn `sPattern` leer ist, wird immer nur das `TCVN_OMT_NUMBERS_SC_UCLETTERS` Model verwendet. Wenn dieses Modell nicht übergeben wurde, wird ein `HRESULT = ADSERR_DEVICE_INVALIDPARM` zurückgegeben.

Beispiele für die Verwendung von sPattern

Bei der Verwendung der optionalen Parameter `sPattern` und `eOcrOptions` ergeben sich zusätzliche Möglichkeiten. Je nach Kombination der Zeichen auf dem Eingangsbild, den Angaben von `sPattern` und `eOcrOptions` und den erkannten Zeichen, wird bei einer Nicht-Übereinstimmung `S_FALSE` zurückgegeben. Danach können die Zeichen des Ergebnis-Strings (`ipCharacters`) auch einzeln analysiert werden, da z.B. bei abweichender Länge auch `S_FALSE` zurückgegeben wird. Die Beispiele beziehen sich auf die Verwendung des `OMT_NUMBERS_SC_UCLETTERS` Modells.

Zeichen auf dem ipSrcImage	sPattern	eOcrOptions	ipCharacters	HRESULT
12/34	dd.dd		12/34	S_OK
12534	dd.dd		12534	S_OK
12/34	dd!dd		1234	S_OK
12 34	dd!d		124	S_OK
12534	dd!dd		1234	S_OK
12 34	dd_dd		12 34	S_OK
AB12/	uudd#		AB12/	S_OK
12 34	dddd	WITHBLANKS	12 34	S_OK
12/4	dd#d	WITHBLANKS	12/4	S_OK
12 34	dd_dd	WITHBLANKS	12 34	S_OK
12 34 56	dd_dddd	WITHBLANKS	12 34 56	S_OK
12 3 4	dd.dd	WITHBLANKS	12 3 4	S_FALSE
12 34	dd!dd		124	S_FALSE
12 34	dd.dd		1234	S_FALSE
1234	!dddd		234	S_FALSE
12 34	dd_dd		12.34	S_FALSE
12 3	dd_dd		123	S_FALSE
12 3 4	dd_d_d		123 4	S_FALSE
AB12/	uddd#		AB12/	S_FALSE
AB12/	uuddd		AB12/	S_FALSE

6.2.3.23.1 OCR

Detects and recognizes characters in a binary image (white characters on black background).
Can use available TwinCAT Job Tasks for executing parallel code regions.

Syntax

Definition:

```
HRESULT OCR(
    HRESULT          hrPrev,
    ITcVnImage*      ipSrcImage,
    ULONGLONG        eModel,
    ITcVnContainer*& ipCharacters,
    float&           fMinConfidence,
    PCCH             sPattern = "",
    ULONG            eOcrOptions = OO_NONE,
    ITcVnContainer** pipBoundingBoxes = nullptr,
    ITcVnContainer** pipConfidences = nullptr
)
```

Parameters

Name	Type	Default	Description
hrPrev	HRESULT [▶ 150]		HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipSrcImage	ITcVnImage* [▶ 1925]		Source image (USINT, 1 channel, binary)
eModel	ULONGLONG		Selection of OCR model types (ETcVnOcrModelType)
ipCharacters	ITcVnContainer* [▶ 1890]&		Returns the recognized characters (ContainerType_Vector_String_SINT)
fMinConfidence	float&		Returns the minimum value of the confidences
sPattern	PCCH	""	String pattern containing the format in which characters are presented
eOcrOptions	ULONG	OO_NONE	Specifies which options must be applied to the function (ETcVnOcrOptions)
pipBoundingBoxes	ITcVnContainer* [▶ 1890]*	nullptr	Returns the corresponding bounding boxes of the recognized characters (ContainerType_Vector_TcVnRectangle_DINT, optional, set to 0 if not required)
pipConfidences	ITcVnContainer* [▶ 1890]*	nullptr	Returns the corresponding classification confidences of the recognized characters (ContainerType_Vector_REAL, optional, set to 0 if not required)

Return value

HRESULT [▶ 150]

Required License

TC3 Vision OCR

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.24 Texture Analysis

Diese Gruppe enthält Funktionen zur Texturanalyse eines Bildes.

Funktionen

Granulometrische Eigenschaften

- Granulometry [▶ 2730]

Haralick Merkmale

- HaralickFeatures [▶ 2731]

6.2.3.24.1 Granulometry

Computes the granulometry of a textured image using morphological operations.

Syntax

Definition:

```
HRESULT Granulometry(
    HRESULT                    hrPrev,
    ITcVnImage*                ipImage,
    ITcVnContainer*&           ipIntensityDiffs,
    ETcVnMorphologicalOperator eMorphType,
    ULONG                      nMinSize,
    ULONG                      nMaxSize,
    ULONG                      nStep
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]	Source image (1 channel)
ipIntensityDiffs	ITcVnContainer* [▶ 1890]&	Returns the intensity differences between successive structuring element sizes (CTcVnContainer_Vector_LREAL), which can be used as features for texture description.
eMorphType	ETcVnMorphologicalOperator [▶ 1746]	Only opening or closing are supported.
nMinSize	ULONG	Min structuring element size (odd, >= 3)
nMaxSize	ULONG	Max structuring element size (>= nMinSize)
nStep	ULONG	Step, by which nMinSize is incremented until nMaxSize is exceeded (even, >= 2).

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.3.24.2 HaralickFeatures

Computes the Haralick features of a graylevel image, which describe the texture.

Syntax

Definition:

```
HRESULT HaralickFeatures(
    HRESULT          hrPrev,
    ITcVnImage*      ipImage,
    ITcVnContainer*& ipFeatures,
    ULONG            nDist
)
```

Parameters

Name	Type	Description
hrPrev	HRESULT [▶ 150]	HRESULT indicating the result of previous operations (If SUCCEEDED(hrPrev) equals false, no operation is executed.)
ipImage	ITcVnImage* [▶ 1925]	Source image (1 channel, USINT)
ipFeatures	ITcVnContainer* [▶ 1890]&	Returns the computed Haralick features (CTcVnContainer_Vector_LREAL). The returned feature set contains the mean and standard deviation for each of the first 13 Haralick features, computed for 0, 45, 90, 135 degrees. So there will be 26 features in total (f1 mean, f1 stdDev, f2 mean, f2 stdDev, ..., f13 mean, f13 stdDev).
nDist	ULONG	Distance

Return value

HRESULT [▶ 150]

Required License

TC3 Machine Learning Realtime Inference

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4 Function Blocks

Die Kommunikation mit GigE Vision Kameras oder mit dem Dateisystem sowie komplexeren Bildverarbeitungsfunktionen wird durch Funktionsblöcke abgebildet.

Gruppierung von Funktionsblöcken

Alle Funktionsblöcke der TwinCAT Vision Bibliothek sind thematisch in folgenden Gruppen sortiert:

- Camera Register Access [▶ 2733]
- File Access [▶ 2744]
- Image Acquisition [▶ 2760]
- Image Processing [▶ 2779]

6.2.4.1 License Overview

Folgend ist die Zugehörigkeit der TwinCAT Vision Funktionsblöcke zu den verfügbaren Lizenzen (siehe Lizenzmodell [▶ 12]) aufgelistet. Daraus ist ersichtlich, welche Lizenz für die Anwendung bestimmter Funktionsblöcke benötigt wird.

Die Basis-Lizenz TC3 Vision Base (TF7100) wird immer benötigt.

TC3 Vision Base

FileSourceControl [▶ 2762]

GevCameraControl [▶ 2765]

InitializeFunction [▶ 2747]

ReadCalibrationPattern [▶ 2748]

ReadCalibrationResult [▶ 2749]

ReadContainer [▶ 2751]

ReadImage [▶ 2752]

ReadMemory [▶ 2735]

ReadMlModel [▶ 2753]

ReadNeuralNetwork [▶ 2759]

ReadRegister_REAL [▶ 2736]

ReadRegister_UDINT [▶ 2737]

ReadRegister_ULINT [▶ 2738]

SimpleCameraControl [▶ 2776]

WriteCalibrationResult [▶ 2754]

WriteContainer [▶ 2755]

WriteImage [▶ 2757]

WriteMemory [▶ 2739]

WriteMlModel [▶ 2758]

WriteRegister_REAL [▶ 2741]

WriteRegister_UDINT [▶ 2742]

WriteRegister_ULINT [▶ 2743]

TC3 Vision Matching

GeneralizedHoughBallard [▶ 2779]

SSIM [▶ 2782]

6.2.4.2 Camera Register Access

Diese Gruppe enthält Funktionsblöcke zum Lesen und Schreiben von Kamera-Registern oder Speicherbereichen.

Funktionsblöcke

UDINT-Register (4 Byte)

- ReadRegister_UDINT [▶ 2737]
- WriteRegister_UDINT [▶ 2742]

ULINT-Register (8 Byte)

- ReadRegister_ULINT [▶ 2738]
- WriteRegister_ULINT [▶ 2743]

REAL-Register (4 Byte)

- ReadRegister_REAL [▶ 2736]
- WriteRegister_REAL [▶ 2741]

Größere Speicherbereiche

- ReadMemory [▶ 2735]
- WriteMemory [▶ 2739]

Parameter

Die folgenden Parameter funktionieren bei allen Funktionsblöcken dieser Gruppe gleich:

Adresse

Die Adresse `nAddress` gibt das zu lesende oder zu speichernde Register der Kamera an. Der Typ GVCP_REGISTER_ADDRESS ist lediglich ein Alias auf UDINT; sodass die Adresse 4 Byte (bzw. 32 Bit) lang sein muss. Bei größeren Speicherbereichen der Kamera (siehe ReadMemory [▶ 2735]) zeigt die Adresse auf den Anfang des zu lesenden oder zu speichernden Bereichs.

Funktion ausführen

Die Trigger für Schreib- und Lese-Befehle sind `bWrite` und `bRead`. Sie werden durch eine steigende Flanke (Wechsel von `false` nach `true`) ausgelöst. Dabei werden die relevanten Eingangsparameter intern gespeichert, sodass eine Änderung der Parameter während der Ausführung keinen Einfluss hat. Ebenso hat eine fallende Flanke der Trigger keine Auswirkung.

> **Funktionsblöcke aufrufen**
>
> Beachten Sie, dass das Beschreiben von Parametern bei Funktionsblöcken nur eine Auswirkung hat, wenn dieser auch aufgerufen wird.

Timeout

Um in unbekannten Fehlerfällen nicht ewig auf die Ausführung des Funktionsblocks zu warten, kann mit `nTimeout` die maximal erlaubte Ausführungszeit festgelegt werden. Wenn diese Zeit ohne erfolgreiches Ergebnis nach Auslösen des Befehlstriggers überschritten wird, wird die Ausführung abgebrochen. Der Timeout-Fehler ist dann über die beiden Ausgänge `bError` und `nErrorId` zu erkennen.

Arbeitsstatus

Der Ausgang `bBusy` gibt an, ob der Funktionsblock beschäftigt ist. Wenn `bBusy == true` kann kein neuer Befehl ausgelöst werden.

Mittels einer geeigneten `if`-Abfrage kann festgestellt werden, ob die Ausführung des Funktionsblocks abgeschlossen ist:

```
if (!fbReadOrWrite.bBusy)
{
    // Function block execution is complete or error occurred.
    // Next write-command can be issued.
}
```

Fehlerstatus

Der Fehlerstatus wird angegeben, indem `bError` binär auf einen Fehler hinweist und `nErrorId` ggfs. den Fehlercode (ADS Return Code [▶ 2966]) ausgibt. Der Fehlerstatus wird zurückgesetzt, sobald ein neuer Befehl ausgelöst wird. Der Fehlercode ist `PENDING (0x71E)`, wenn der Funktionsblock beschäftigt ist (`bBusy == true`).

Der Hexadezimalwert des Fehlercodes kann wie folgt extrahiert werden:

```
if (fbReadOrWrite.bError)
{
    // Show relevant error code for debugging purposes:
    nErrorCode = fbReadOrWrite.nErrorId & 0xFFF;
}
```

6.2.4.2.1 ReadMemory

This FB reads consecutive memory locations from the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before)

Syntax

Constructor:

```
ReadMemory(
    OTCID oidITcVnGevImageProvider
)
```

Execute main:

```
FB_Main()

FB_Main(
    GVCP_REGISTER_ADDRESS nAddress,
    USHORT                nLength,
    PVOID                 pBuffer,
    bool                  bRead
)

FB_Main(
    bool bRead
)
```

Inputs

Name	Type	Default	Description
nAddress	GVCP_REGISTER_ADDRESS [▶ 1711]		The address of the first byte to read from the camera (must be 32-bit aligned)
nLength	USHORT		The number of bytes to read, starting with nAddress (must be a multiple of 4, not more than 536)
pBuffer	PVOID		Pointer to the buffer where the read memory content is written to (The buffer must have a size of at least nLength bytes!)
bRead	bool		Reading the memory is triggered by a rising edge at this input.
nTimeout	unsigned int	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	bool	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	bool	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	ULONG	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.2.2 ReadRegister_REAL

This FB reads a REAL register from the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before)

Syntax

Constructor:

```
ReadRegister_REAL(
    OTCID oidITcVnGevImageProvider
)
```

Execute main:

```
FB_Main()

FB_Main(
    GVCP_REGISTER_ADDRESS nAddress,
    bool                  bRead
)

FB_Main(
    bool bRead
)
```

Inputs

Name	Type	Default	Description
nAddress	GVCP_REGISTER_ADDRESS [▶ 1711]		The address of the register that should be read
nEndian	USHORT		OPTIONAL: The endianness of the register. 0 = Big, 1 = Little. Default: 0
bRead	bool		Reading the register is triggered by a rising edge at this input.
nTimeout	unsigned int	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
fValue	float	The read value
bBusy	bool	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	bool	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	ULONG	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.2.3 ReadRegister_UDINT

This FB reads a DINT register from the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before)

Syntax

Constructor:

```
ReadRegister_UDINT(
    OTCID oidITcVnGevImageProvider
)
```

Execute main:

```
FB_Main()

FB_Main(
    GVCP_REGISTER_ADDRESS nAddress,
    bool                  bRead
```

```
)

FB_Main(
    bool bRead
)
```

Inputs

Name	Type	Default	Description
nAddress	GVCP_REGISTER_ADDRESS [▶ 1711]		The address of the register that should be read
nEndian	USHORT		OPTIONAL: The endianness of the register. 0 = Big, 1 = Little. Default: 0
bRead	bool		Reading the register is triggered by a rising edge at this input.
nTimeout	unsigned int	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
nValue	ULONG	The read value
bBusy	bool	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	bool	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	ULONG	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.2.4 ReadRegister_ULINT

This FB reads a ULINT register from the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before)

Syntax

Constructor:

```
ReadRegister_ULINT(
    OTCID oidITcVnGevImageProvider
)
```

Execute main:

```
FB_Main()

FB_Main(
    GVCP_REGISTER_ADDRESS nAddress,
    bool                  bRead
)

FB_Main(
    bool bRead
)
```

Inputs

Name	Type	Default	Description
nAddress	GVCP_REGISTER_ADDRESS [▶ 1711]		The address of the register that should be read
nEndian	USHORT		OPTIONAL: The endianness of the register. 0 = Big, 1 = Little. Default: 0
bRead	bool		Reading the register is triggered by a rising edge at this input.
nTimeout	unsigned int	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
nValue	ULONGLONG	The read value
bBusy	bool	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	bool	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	ULONG	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.2.5 WriteMemory

This FB writes data to consecutive memory locations on the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before)

Syntax

Constructor:

```
WriteMemory(
    OTCID oidITcVnGevImageProvider
)
```

Execute main:

```
FB_Main()

FB_Main(
    GVCP_REGISTER_ADDRESS nAddress,
    USHORT                nLength,
    PVOID                 pData,
    bool                  bWrite
)

FB_Main(
    bool bWrite
)
```

Inputs

Name	Type	Default	Description
nAddress	GVCP_REGISTER_ADDRESS [▶ 1711]		The address of the first byte to write to the camera (must be 32-bit aligned)
nLength	USHORT		The number of bytes to write, starting at nAddress (must be a multiple of 4, not more than 536)
pData	PVOID		Pointer to the data that should be written to the camera (must have a size of at least nLength bytes!)
bWrite	bool		Writing the memory is triggered by a rising edge at this input.
nTimeout	unsigned int	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	bool	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	bool	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	ULONG	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.
nBytesWritten	USHORT	The number of bytes that have been successfully written to the camera.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.2.6 WriteRegister_REAL

This FB writes a REAL value into a register on the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before)

Syntax

Constructor:

```
WriteRegister_REAL(
    OTCID oidITcVnGevImageProvider
)
```

Execute main:

```
FB_Main()

FB_Main(
    GVCP_REGISTER_ADDRESS nAddress,
    float                 fValue,
    bool                  bWrite
)

FB_Main(
    bool bWrite
)
```

Inputs

Name	Type	Default	Description
nAddress	GVCP_REGISTER_ADDRESS [▶ 1711]		The address of the register that should be written
fValue	float		The value to write
nEndian	USHORT		OPTIONAL: The endianness of the register. 0 = Big, 1 = Little. Default: 0
bWrite	bool		Writing the register is triggered by a rising edge at this input.
nTimeout	unsigned int	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	bool	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	bool	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	ULONG	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.2.7 WriteRegister_UDINT

This FB writes a DINT value into a register on the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before)

Syntax

Constructor:

```
WriteRegister_UDINT(
    OTCID oidITcVnGevImageProvider
)
```

Execute main:

```
FB_Main()

FB_Main(
    GVCP_REGISTER_ADDRESS nAddress,
    ULONG                 nValue,
    bool                  bWrite
)

FB_Main(
    bool bWrite
)
```

Inputs

Name	Type	Default	Description
nAddress	GVCP_REGISTER_ADDRESS [▶ 1711]		The address of the register that should be written
nValue	ULONG		The value to write
nEndian	USHORT		OPTIONAL: The endianness of the register. 0 = Big, 1 = Little. Default: 0
bWrite	bool		Writing the register is triggered by a rising edge at this input.
nTimeout	unsigned int	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	bool	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	bool	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	ULONG	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.2.8 WriteRegister_ULINT

This FB writes a ULINT value into a register on the camera. Requires an open control channel (e.g. by calling FB_VN_GevCameraControl.OpenCamera() before)

Syntax

Constructor:

```
WriteRegister_ULINT(
    OTCID oidITcVnGevImageProvider
)
```

Execute main:

```
FB_Main()

FB_Main(
    GVCP_REGISTER_ADDRESS nAddress,
    ULONGLONG             nValue,
```

```
    bool                    bWrite
)

FB_Main(
    bool bWrite
)
```

Inputs

Name	Type	Default	Description
nAddress	GVCP_REGISTER_ADDRESS [▶ 1711]		The address of the register that should be written
nValue	ULONGLONG		The value to write
nEndian	USHORT		OPTIONAL: The endianness of the register. 0 = Big, 1 = Little. Default: 0
bWrite	bool		Writing the register is triggered by a rising edge at this input.
nTimeout	unsigned int	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	bool	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	bool	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	ULONG	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.3 File Access

Diese Gruppe enthält Funktionsblöcke zum Lesen und Schreiben von Daten auf dem Dateisystem.

Funktionsblöcke

Bilder

- ReadImage [▶ 2752]
- WriteImage [▶ 2757]

Container

- ReadContainer [▶ 2751]
- WriteContainer [▶ 2755]

Machine learning Modelle

- ReadMlModel [▶ 2753]
- WriteMlModel [▶ 2758]

Kalibrationsmuster

- ReadCalibrationPattern [▶ 2748]
- ReadCalibrationResult [▶ 2749]
- WriteCalibrationResult [▶ 2754]

Funktionsinitialisierung

- InitializeFunction [▶ 2747]

Neural Network

- ReadNeuralNetwork [▶ 2759]

Allgemeines

Diese Funktionsblöcke arbeiten asynchron; die Zeit von der ersten Ausführung bis zum gewünschten Ergebnis erstreckt sich über mehrere Zyklen. Die tatsächliche Dauer jedes Lese-/Schreibvorgangs hängt von verschiedenen Faktoren ab, z.B. von der Größe des Bildes und dem Verhalten des Betriebssystems. Die folgenden Parameter sind bei allen Funktionsblöcken gleich und sind daher einmal hier zentral beschrieben.

Parameter

Dateipfad

Der Dateipfad `sFilePath` legt fest, wo eine Datei gespeichert werden soll bzw. welche Datei geladen werden soll. Es können absolute und relative Pfade verwendet werden. Bei relativen Pfaden werden die Standardpfade, die in der Service Configuration [▶ 68] festgelegt sind, als Basis verwendet.

Der Dateipfad wird bei einer steigenden Flanke des Schreib-Befehls `bWrite` intern gespeichert, sodass Änderungen des Pfads in späteren Zyklen keine Auswirkung haben.

Funktion ausführen

Die Trigger für Schreib- und Lese-Befehle sind `bWrite` und `bRead`. Sie werden durch eine steigende Flanke (Wechsel von `false` nach `true`) ausgelöst. Dabei werden die relevanten Eingangsparameter intern gespeichert, sodass eine Änderung der Parameter während der Ausführung keinen Einfluss hat. Ebenso hat eine fallende Flanke der Trigger keine Auswirkung.

> **Funktionsblöcke aufrufen**
>
> Beachten Sie, dass das Beschreiben von Parametern bei Funktionsblöcken nur eine Auswirkung hat, wenn dieser auch aufgerufen wird.

Timeout

Um in unbekannten Fehlerfällen nicht ewig auf die Ausführung des Funktionsblocks zu warten, kann mit `nTimeout` die maximal erlaubte Ausführungszeit festgelegt werden. Wenn diese Zeit ohne erfolgreiches Ergebnis nach Auslösen des Befehlstriggers überschritten wird, wird die Ausführung abgebrochen. Der Timeout-Fehler ist dann über die beiden Ausgänge `bError` und `nErrorId` zu erkennen.

Arbeitsstatus

Der Ausgang `bBusy` gibt an, ob der Funktionsblock beschäftigt ist. Wenn `bBusy == true` kann kein neuer Befehl ausgelöst werden.

Mittels einer geeigneten `if`-Abfrage kann festgestellt werden, ob die Ausführung des Funktionsblocks abgeschlossen ist:

```
if (!fbReadOrWrite.bBusy)
{
    // Function block execution is complete or error occurred.
    // Next write-command can be issued.
}
```

Fehlerstatus

Der Fehlerstatus wird angegeben, indem `bError` binär auf einen Fehler hinweist und `nErrorId` ggfs. den Fehlercode (ADS Return Code [▶ 2966]) ausgibt. Der Fehlerstatus wird zurückgesetzt, sobald ein neuer Befehl ausgelöst wird. Der Fehlercode ist `PENDING (0x71E)`, wenn der Funktionsblock beschäftigt ist (`bBusy == true`).

Der Hexadezimalwert des Fehlercodes kann wie folgt extrahiert werden:

```
if (fbReadOrWrite.bError)
{
    // Show relevant error code for debugging purposes:
    nErrorCode = fbReadOrWrite.nErrorId & 0xFFF;
}
```

Die häufigsten Fehlercodes sind:

Hex	Dec	Name	Bedeutung
0x006	6	TARGETPORTNOTFOUND	ADS-Server nicht erreichbar. Tritt im Wesentlichen auf, wenn der TwinCAT Vision Service nicht gestartet ist. Siehe Vision Service startet nicht [▶ 2951].
0x700	1792	ERROR	Allgemeiner Fehler, wenn das Speichern oder Laden fehlschlägt. Dieser weist meistens darauf hin, dass das angegebene Dateiformat nicht unterstützt wird oder, dass der Dateipfad nicht erreichbar ist. Dieser Fehler tritt im Regelfall im TwinCAT Vision Service [▶ 68] auf.
0x70B	1803	INVALIDPARM	Invalider Parameter. Dies weist meistens darauf hin, dass der Interface Pointer [▶ 146] nicht valide ist.
0x719	1817	TIMEOUT	Timeout. Das Laden oder Speichern hat die maximale Ausführungszeit `nTimeout` überschritten. Generell tritt dies hauptsächlich bei starker Auslastung von ADS auf.
0x71E	1822	PENDING	Kein wirklicher Fehler, daher ist hierbei auch der Ausgang `bError` nicht gesetzt. Dieser Code signalisiert, dass der Funktionsblock beschäftigt ist. Er tritt parallel zum Ausgang `bBusy` auf, falls während Beschäftigtseins des Funktionsblocks eine neue steigende Flanke registriert wird.

6.2.4.3.1 InitializeFunction

This FB initializes selected functions.

Syntax

Constructor:

```
InitializeFunction()
```

Execute main:

```
FB_Main()

FB_Main(
    ETcVnInitializableFunction eFunction,
    ULONGLONG                  nOptions,
```

```
    bool                        bStart
)

FB_Main(
    bool bStart
)
```

Inputs

Name	Type	Default	Description
eFunction	ETcVnInitializableFunction [▶ 1743]		Selects the function to initialize
nOptions	ULONGLONG	0	Selects initialization options for the function
bStart	bool		Function initialization is triggered by a rising edge at this input.
nTimeout	unsigned int	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	bool	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	bool	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	ULONG	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.3.2 ReadCalibrationPattern

This FB reads calibration pattern reference points from an xml file on the targte pc.

Syntax

Constructor:

```
ReadCalibrationPattern()
```

Execute main:

```
FB_Main()

FB_Main(
    PCCH             sFilePath,
    ITcVnContainer** pipDestContainer,
    bool             bRead
)

FB_Main(
    bool bRead
)
```

Inputs

Name	Type	Default	Description
sFilePath	PCCH		Full path of the file or relative path to the default directory on the target pc
pipDestContainer	ITcVnContainer* [▶ 1890]*		Returns a container with the pattern points (ContainerType_Vector_TcVnPoint3_REAL)
bRead	bool		Reading the file is triggered by a rising edge at this input.
nTimeout	unsigned int	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	bool	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	bool	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	ULONG	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.3.3 ReadCalibrationResult

This FB reads calibration results from a json file on the targte pc.

Syntax

Constructor:

```
ReadCalibrationResult()
```

Execute main:

```
FB_Main()

FB_Main(
    PCCH                  sFilePath,
    TcVnMatrix3x3_LREAL*  paCameraMatrix,
    TcVnArray8_LREAL*     paDistortionCoefficients,
    TcVnMatrix3x3_LREAL*  paRotationMatrix,
    TcVnVector3_LREAL*    paTranslationVector,
    bool                  bRead
)

FB_Main(
    bool bRead
)
```

Inputs

Name	Type	Default	Description
sFilePath	PCCH		Full path of the file or relative path to the default directory on the target pc
paCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]*		Returns the camera matrix
paDistortionCoefficients	TcVnArray8_LREAL [▶ 1713]*		Returns the distortion coefficients
paRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]*		Returns the rotation matrix
paTranslationVector	TcVnVector3_LREAL [▶ 1713]*		Returns the translation vector
bRead	bool		Reading the file is triggered by a rising edge at this input.
nTimeout	unsigned int	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	bool	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	bool	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	ULONG	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.3.4 ReadContainer

This FB reads a container from an xml file on the target pc.

Syntax

Constructor:

```
ReadContainer()
```

Execute main:

```
FB_Main()

FB_Main(
    PCCH             sFilePath,
    ITcVnContainer** pipDestContainer,
    bool             bRead
)

FB_Main(
    PCCH             sFilePath,
    ITcVnContainer** pipDestContainer,
    GUID             nDestTypeGuid,
    bool             bRead
)

FB_Main(
    bool bRead
)
```

Inputs

Name	Type	Default	Description
sFilePath	PCCH		Full path of the file or relative path to the default directory on the target pc
pipDestContainer	ITcVnContainer* [▶ 1890]*		Returns a container with the loaded content
nDestTypeGuid	GUID		The type id of the container to return. If this parameter is not set, a suitable container type will be determined automatically (except for csv files).
bRead	bool		Reading the file is triggered by a rising edge at this input.
nTimeout	unsigned int	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	bool	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	bool	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	ULONG	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.3.5 ReadImage

This FB reads an image from a file on the target pc.

Syntax

Constructor:

```
ReadImage()
```

Execute main:

```
FB_Main()

FB_Main(
    PCCH         sFilePath,
    ITcVnImage** pipDestImage,
    bool         bRead
)

FB_Main(
    bool bRead
)
```

Inputs

Name	Type	Default	Description
sFilePath	PCCH		Full path of the file or relative path to the default image directory on the target pc
pipDestImage	ITcVnImage* [▶ 1925]*		Returns an image with the loaded content
bRead	bool		Reading the file is triggered by a rising edge at this input.
nTimeout	unsigned int	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	bool	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	bool	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	ULONG	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.3.6 ReadMlModel

This FB reads a machine learning model from a file on the target pc.

Syntax

Constructor:

```
ReadMlModel()
```

Execute main:

```
FB_Main()

FB_Main(
    PCCH            sFilePath,
    ITcVnMlModel**  pipDestMlModel,
    bool            bRead
)

FB_Main(
    bool bRead
)
```

Inputs

Name	Type	Default	Description
sFilePath	PCCH		Full path of the file or relative path to the default directory on the target pc
pipDestMlModel	ITcVnMlModel* [▶ 1933]*		Returns the loaded machine learning model
bRead	bool		Reading the file is triggered by a rising edge at this input.
nTimeout	unsigned int	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	bool	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	bool	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	ULONG	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.3.7 WriteCalibrationResult

This FB writes calibration results as a json file to the hard drive.

Syntax

Constructor:

```
WriteCalibrationResult()
```

Execute main:

```
FB_Main()

FB_Main(
    PCCH                 sFilePath,
    TcVnMatrix3x3_LREAL* paCameraMatrix,
    TcVnArray8_LREAL*    paDistortionCoefficients,
    TcVnMatrix3x3_LREAL* paRotationMatrix,
    TcVnVector3_LREAL*   paTranslationVector,
    bool                 bWrite
)
```

```
FB_Main(
    bool bWrite
)
```

Inputs

Name	Type	Default	Description
sFilePath	PCCH		Full path of the file or relative path to the default directory on the target pc (e.g. only a file name). If the string is empty, a file name will be generated (containing the current time).
paCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]*		Returns the camera matrix
paDistortionCoefficients	TcVnArray8_LREAL [▶ 1713]*		Returns the distortion coefficients
paRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]*		Returns the rotation matrix
paTranslationVector	TcVnVector3_LREAL [▶ 1713]*		Returns the translation vector
bWrite	bool		Writing the container is triggered by a rising edge at this input.
nTimeout	unsigned int	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	bool	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	bool	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	ULONG	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.3.8 WriteContainer

This FB writes a container as an xml file to the hard drive.

Syntax

Constructor:

```
WriteContainer()
```

Execute main:

```
FB_Main()

FB_Main(
    ITcVnContainer**           pipContainer,
    PCCH                       sFilePath,
    ETcVnContainerExportFormat eExportFormat,
    bool                       bWrite
)

FB_Main(
    bool bWrite
)
```

Inputs

Name	Type	Default	Description
pipContainer	ITcVnContainer* [▶ 1890]*		The container to write
sFilePath	PCCH		Full path of the file or relative path to the default directory on the target pc (e.g. only a file name). If the string is empty, a file name will be generated (containing the current time).
eExportFormat	ETcVnContainerExportFormat [▶ 1733]	CEF_XML	Container export format (default is human readable xml)
bWrite	bool		Writing the container is triggered by a rising edge at this input.
nTimeout	unsigned int	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	bool	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	bool	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	ULONG	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.3.9 WriteImage

This FB writes an image to the hard drive.

Syntax

Constructor:

```
WriteImage()
```

Execute main:

```
FB_Main()

FB_Main(
    ITcVnImage** pipImage,
    PCCH         sFilePath,
    bool         bWrite
)

FB_Main(
    bool bWrite
)
```

Inputs

Name	Type	Default	Description
pipImage	ITcVnImage* [▶ 1925]*		The image to write
sFilePath	PCCH		Full path of the file or relative path to the default directory on the target pc (e.g. only a file name). If the string is empty, a file name will be generated (containing the current time).
bWrite	bool		Writing the image is triggered by a rising edge at this input.
nTimeout	unsigned int	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	bool	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	bool	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	ULONG	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.3.10 WriteMlModel

This FB writes a machine learning model to the hard drive.

Syntax

Constructor:

```
WriteMlModel()
```

Execute main:

```
FB_Main()

FB_Main(
    ITcVnMlModel** pipMlModel,
    PCCH           sFilePath,
    bool           bWrite
)

FB_Main(
    bool bWrite
)
```

Inputs

Name	Type	Default	Description
pipMlModel	ITcVnMlModel* [▶ 1933]*		The machine learning model to write
sFilePath	PCCH		Full path of the file or relative path to the default directory on the target pc (e.g. only a file name). If the string is empty, a file name will be generated (containing the current time).
bWrite	bool		Writing the machine learning model is triggered by a rising edge at this input.
nTimeout	unsigned int	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	bool	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	bool	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	ULONG	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.3.11 ReadNeuralNetwork

This FB reads a neural network from a file on the target pc.

Syntax

Constructor:

```
ReadNeuralNetwork()
```

Execute main:

```
FB_Main()

FB_Main(
    PCCH                  sFilePath,
    ITcVnNeuralNetwork**  pipDestNeuralNetwork,
    bool                  bRead
)

FB_Main(
    PCCH                  sFilePath,
    ITcVnNeuralNetwork**  pipDestNeuralNetwork,
    TcVnArray33_UDINT*    paInputShape,
    bool                  bRead
)

FB_Main(
    bool bRead
)
```

Inputs

Name	Type	Default	Description
sFilePath	PCCH		Full path of the file or relative path to the default directory on the target pc
pipDestNeuralNetwork	ITcVnNeuralNetwork* [▶ 1941]*		Returns the neural network
paInputShape	TcVnArray33_UDINT [▶ 1713]*		Optional input shape for testing the execution of the neural network model. The error code INCOMPATIBLE is returned if the execution fails. Setting aInputShape = 0 or aInputShape[0] = 0 disables the testing of the execution.
bRead	bool		Reading the file is triggered by a rising edge at this input.
nTimeout	unsigned int	VISION_ADS_TIMEOUT	Indicates the time before the function is cancelled.

Outputs

Name	Type	Description
bBusy	bool	This output remains TRUE until the function block has executed a command, but at the longest for the duration supplied to the 'nTimeout' input. While bBusy = TRUE, no new command will be accepted at the inputs.
bError	bool	This output is switched to TRUE as soon as an error occurs during the execution of a command. The command-specific error code is contained in ‘nErrorId’. If the function block has a timeout error, 'bError' is TRUE and 'nErrorId' is 1861 (hexadecimal 0x745). Is reset to FALSE by the execution of a command at the inputs.
nErrorId	ULONG	Contains the ADS error code or the command-specific error code of the last executed command. Is reset to 0 by the execution of a command at the inputs.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.4 Image Acquisition

Diese Gruppe enthält Funktionsblöcke zur Bildaufnahme mit Vision-Geräten (GigE Vision Kamera [▶ 72] und File Source [▶ 135] Objekte).

Funktionsblöcke

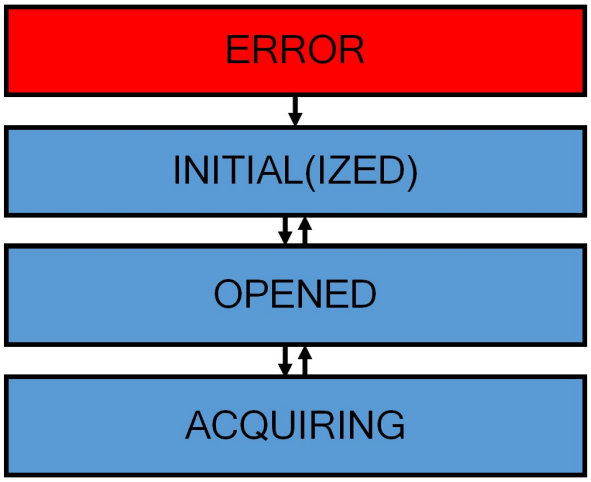
- SimpleCameraControl [▶ 2776] für Kameras und File Sources
- GevCameraControl [▶ 2765] für GigE Vision Kameras
- FileSourceControl [▶ 2762] für File Sources

Allgemeines

Die Funktionsblöcke in dieser Gruppe stellen einen Proxy für die Image Provider TcCOM-Objekte dar. Dementsprechend müssen Sie nicht zyklisch aufgerufen werden. Stattdessen wird das TcCOM-Objekt über die Methoden des Funktionsblocks gesteuert.

Zustandsautomat

Alle Funktionsblöcke zur Bildaufnahme beinhalten einen Zustandsautomaten, um den Zustand des repräsentierten Geräts zu verwalten. Im Wesentlichen muss ein Funktionsblock in den ACQUIRING Zustand gebracht werden, um Bilder kontinuierlich zu empfangen. Der INITIAL Zustand stellt den Ausgangspunkt nach einem Neustart oder Reset dar, von dem aus das INITIALIZING angestoßen werden kann. Ab dem OPENED-Zustand ist der Kommunikationskanal zu dem Gerät geöffnet, um z.B. Registerwerte auslesen oder schreiben zu können.



Alle Zustände werden durch das Enum ETcVnCameraState [▶ 1718] repräsentiert und sind folgend erklärt:

Hauptzustände

Die Hauptzustände sind stationäre Zustände. Sie stellen im Wesentlichen dar, ob Verbindung zu einem Vision-Geräte besteht (`OPENED`) und ob das Gerät Bilder aufnimmt (`ACQUIRING`) sowie ob es einen Fehler bei der Verbindung mit dem Gerät gibt (`ERROR`). Zwischen diesen Zuständen können Sie mit den Methoden der FBs wechseln. Im Regelfall erfolgt der Übergang zwischen zwei Hauptzuständen nicht sofort, sondern über mindestens einen Zyklus. In dieser Zwischenzeit ist das Vision-Gerät in einem Zwischenzustand [▶ 2761].

Zustand	Beschreibung
ERROR	Fehlerzustand, aus dem jedes Gerät nur durch Aufruf der Methode `Reset` reaktiviert werden kann.
INITIAL	Initialer Zustand jedes Geräts.
INITIALIZED	Ähnlich wie INITIAL, wobei die Kamera manuell initialisiert wurde (z.B. mit Force-IP und den Initialization Commands)
OPENED	Es besteht eine Verbindung zur Kamera, einzelne Bilder können getriggert werden und Kameraregister können gelesen und beschrieben werden. **Dieser Zustand kann mit der Methode StartAcquisition übersprungen werden.**
ACQUIRING	Gerät befindet sich im Aufnahmezustand und sendet je nach Einstellung des Geräts Bilder.

Zwischenzustände

Die Zwischenzustände sind nötig, da die meisten Aktionen nicht sofort fertiggestellt werden können (z. B. die Bildaufnahme). Wenn sich die Funktionsblöcke in einem dieser Zwischenzustände befinden, muss die entsprechende Methode erneut aufgerufen werden, um den Übergang in den jeweiligen Hauptzustand zu vervollständigen.

Zustand	Beschreibung
INITIALIZING	INITIAL > INITIALIZED
OPENING	INITIAL > OPENED und INITIALIZED > OPENED
STARTACQUISITION	OPENED > ACQUIRING.
STOPACQUISITION	ACQUIRING > OPENED
RESETTINGFEATURES	OPENED > OPENED
TRIGGERING	OPENED > OPENED und ACQUIRING > ACQUIRING
CLOSING	OPENED > INITIALIZED

Methoden müssen in Zwischenzuständen aufgerufen werden!

Eine jeweilige Methode für Zustandsübergänge muss auch in den Zwischenzuständen aufgerufen werden. Erst wenn nach einem entsprechenden Methodenaufruf ein stationärer Zustand mit der Methode `GetState` festgestellt wird, ist der Zustandsübergang vollständig abgeschlossen und die Methode muss nicht mehr aufgerufen werden. Ein einmaliges Antriggern einer Methode und darauffolgendes Warten führt nicht zu dem gewünschten Zustandsübergang.

Ein Zustandsautomat in C++ kann dann entweder mit einer `switch` Struktur oder mit einer `if … else if …` -Struktur implementiert werden.

6.2.4.4.1 FileSourceControl

This FB provides access to the images send by a File Source instance.

Syntax

Constructor:

```
FileSourceControl(
    OTCID oidITcVnFileImageProvider
)
```

Methods

Name	Description
GetCurrentImage [▶ 2763]	Gets the current available image (if any).
GetCurrentImageAndFileName [▶ 2763]	Gets the current available image (if any) and the corresponding file name.
StartAcquisition [▶ 2763]	Activates the File Source. If not in trigger mode, it will start sending images.
StopAcquisition [▶ 2764]	Deactivates the File Source. It will stop sending images.
Reset [▶ 2764]	Reset the controller to initial state (might require multiple calls depending on current state, until S_OK is returned)
GetState [▶ 2764]	Gets the current state of the internal camera control state machine.
TriggerImage [▶ 2764]	If in trigger mode, this function triggers the next image in the list.
TriggerImageExp [▶ 2764]	If in trigger mode, this function triggers a specific image in the list.
TriggerImageByName [▶ 2765]	If in trigger mode, this function triggers a specific image in the list by name.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.4.1.1 GetCurrentImage

Gets the current available image (if any).

Syntax

Definition:

```
HRESULT GetCurrentImage(
    ITcVnImage*& ipImage
)
```

Parameters

Name	Type	Description
ipImage	ITcVnImage* [▶ 1925]&	Returns the image

Return value

HRESULT [▶ 150]

6.2.4.4.1.2 GetCurrentImageAndFileName

Gets the current available image (if any) and the corresponding file name.

Syntax

Definition:

```
HRESULT GetCurrentImageAndFileName(
    ITcVnImage*& ipImage,
    PCHAR        sFileName,
    USHORT       nMaxLen
)
```

Parameters

Name	Type	Description
ipImage	ITcVnImage* [▶ 1925]&	Returns the image
sFileName	PCHAR	Returns the file name
nMaxLen	USHORT	Define the maximum allowed length for the file name

Return value

HRESULT [▶ 150]

6.2.4.4.1.3 StartAcquisition

Activates the File Source. If not in trigger mode, it will start sending images.

Syntax

Definition:

```
HRESULT StartAcquisition()
```

Return value

HRESULT [▶ 150]

6.2.4.4.1.4 StopAcquisition

Deactivates the File Source. It will stop sending images.

Syntax

Definition:

```
HRESULT StopAcquisition()
```

Return value

HRESULT [▶ 150]

6.2.4.4.1.5 Reset

Reset the controller to initial state (might require multiple calls depending on current state, until S_OK is returned)

Syntax

Definition:

```
HRESULT Reset()
```

Return value

HRESULT [▶ 150]

6.2.4.4.1.6 GetState

Gets the current state of the internal camera control state machine.

Syntax

Definition:

```
ETcVnCameraState GetState()
```

Return value

ETcVnCameraState [▶ 1718]

6.2.4.4.1.7 TriggerImage

If in trigger mode, this function triggers the next image in the list.

Syntax

Definition:

```
HRESULT TriggerImage()
```

Return value

HRESULT [▶ 150]

6.2.4.4.1.8 TriggerImageExp

If in trigger mode, this function triggers a specific image in the list.

Syntax

Definition:

```
HRESULT TriggerImageExp(
    LONG nSkipImages
)
```

Parameters

Name	Type	Description
nSkipImages	LONG	Amount of images to skip, relative to current

Return value

HRESULT [▶ 150]

6.2.4.4.1.9 TriggerImageByName

If in trigger mode, this function triggers a specific image in the list by name.

Syntax

Definition:

```
HRESULT TriggerImageByName(
    PCCH sImageName
)
```

Parameters

Name	Type	Description
sImageName	PCCH	Name of the image to trigger

Return value

HRESULT [▶ 150]

6.2.4.4.2 GevCameraControl

This FB provides the basic functionality to control a GigE Vision camera and access its calibration data.

Syntax

Constructor:

```
GevCameraControl(
    OTCID oidITcVnGevImageProvider
)
```

Methods

Name	Description
ClearImageQueue [▶ 2768]	Delete all images contained in the TcVnGevImageProvider TcCOM module receive queue and reset the corresponding omitted images counter.
CloseCamera [▶ 2768]	Close the control channel to the camera.
GetCalibPatternRef [▶ 2768]	Gets the reference calibration pattern points from the TcVnGevImageProvider TcCOM module (Can be set from the calibration assistant)
GetCameraMatrix [▶ 2769]	Gets the camera matrix from the TcVnGevImageProvider TcCOM module (requires intrinsic calibration parameters, e.g. set by the calibration assistant)
GetCurrentImage [▶ 2769]	Gets the current available image (first in receive queue).
GetCurrentImageUndistorted [▶ 2770]	Gets the current available image (first in receive queue) with undistortion applied. Incompatible with pixel formats that contain an encoding, e.g. a Bayer pattern. Requires intrinsic calibration parameters in the TcVnGevImageProvider TcCOM module, e.g. set by the calibration assistant.
GetCurrentImageWithGvspInfo [▶ 2770]	Gets the current available image (first in receive queue) and its corresponding GVSP info.
GetCurrentImageWithGvspInfo Undistorted [▶ 2770]	Gets the current available image (first in receive queue) with its corresponding GVSP info and undistortion applied. Incompatible with pixel formats that contain an encoding, e.g. a Bayer pattern. Requires intrinsic calibration parameters in the TcVnGevImageProvider TcCOM module, e.g. set by the calibration assistant.
GetCurrentImageWithTimestamps [▶ 2771]	Gets the current available image (first in receive queue) and corresponding timestamps.
GetDistortionCoefficients [▶ 2771]	Gets the distortion coefficients from the TcVnGevImageProvider TcCOM module (requires intrinsic calibration parameters, e.g. set by the calibration assistant)
GetLastImageFromQueue [▶ 2772]	Gets the last received image from the queue.
GetOmittedImagesNum [▶ 2772]	Gets the number of omitted images since the last call of ClearImageQueue(). If the image receive queue in the TcVnGevImageProvider TcCOM module is full and a new image arrives, the first one in the queue will be deleted and the omitted counter is increased.
GetRotationMatrix [▶ 2772]	Gets the rotation matrix from the TcVnGevImageProvider TcCOM module (requires extrinsic calibration parameters, e.g. set by the calibration assistant)
GetState [▶ 2773]	Gets the current state of the internal camera control state machine.
GetTranslationVector [▶ 2773]	Gets the translation vector from the TcVnGevImageProvider TcCOM module (requires extrinsic calibration parameters, e.g. set by the calibration assistant)
InitializeCamera [▶ 2773]	Initialize the camera to the intended state (includes sending the 'InitCameraCommands' defined in the GevImageAcquisition TcCOM module to the camera).
OpenCamera [▶ 2774]	Open a control channel to the camera.
Reset [▶ 2774]	Reset the camera controller to initial state (might require multiple calls depending on current state, until S_OK is returned)
ResetCameraFeatures [▶ 2774]	Reset the camera features to initial state (sends the 'InitCameraCommands' defined in the GevImageAcquisition TcCOM module to the camera).
SetCameraMatrix [▶ 2774]	Sets the camera matrix to the TcVnGevImageProvider TcCOM module
SetDistortionCoefficients [▶ 2775]	Sets the distortion coefficients to the TcVnGevImageProvider TcCOM module

Name	Description
SetRotationMatrix [▶ 2775]	Sets the rotation matrix to the TcVnGevImageProvider TcCOM module
SetTranslationVector [▶ 2775]	Sets the translation vector to the TcVnGevImageProvider TcCOM module
StartAcquisition [▶ 2776]	Send the 'StartAcquisitionCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.
StopAcquisition [▶ 2776]	Send the 'StopAcquisitionCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.
TriggerImage [▶ 2776]	Send the 'SoftwareTriggerCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.4.2.1 ClearImageQueue

Delete all images contained in the TcVnGevImageProvider TcCOM module receive queue and reset the corresponding omitted images counter.

Syntax

Definition:

```
HRESULT ClearImageQueue()
```

Return value

HRESULT [▶ 150]

6.2.4.4.2.2 CloseCamera

Close the control channel to the camera.

Syntax

Definition:

```
HRESULT CloseCamera()
```

Return value

HRESULT [▶ 150]

6.2.4.4.2.3 GetCalibPatternRef

Gets the reference calibration pattern points from the TcVnGevImageProvider TcCOM module (Can be set from the calibration assistant)

Syntax

Definition:

```
HRESULT GetCalibPatternRef(
    ITcVnContainer*& ipCalibPatternRef
)
```

Parameters

Name	Type	Description
ipCalibPatternRef	ITcVnContainer* [▶ 1890]&	Returns the reference calibration pattern points (ContainerType_Vector_TcVnPoint3_REAL)

Return value

HRESULT [▶ 150]

6.2.4.4.2.4 GetCameraMatrix

Gets the camera matrix from the TcVnGevImageProvider TcCOM module (requires intrinsic calibration parameters, e.g. set by the calibration assistant)

Syntax

Definition:

```
HRESULT GetCameraMatrix(
    TcVnMatrix3x3_LREAL& aCameraMatrix
)
```

Parameters

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	Returns the camera matrix

Return value

HRESULT [▶ 150]

6.2.4.4.2.5 GetCurrentImage

Gets the current available image (first in receive queue).

Syntax

Definition:

```
HRESULT GetCurrentImage(
    ITcVnImage*& ipImage
)
```

Parameters

Name	Type	Description
ipImage	ITcVnImage* [▶ 1925]&	Returns the image

Return value

HRESULT [▶ 150]

6.2.4.4.2.6 GetCurrentImageUndistorted

Gets the current available image (first in receive queue) with undistortion applied. Incompatible with pixel formats that contain an encoding, e.g. a Bayer pattern. Requires intrinsic calibration parameters in the TcVnGevImageProvider TcCOM module, e.g. set by the calibration assistant.

Syntax

Definition:

```
HRESULT GetCurrentImageUndistorted(
    ITcVnImage*& ipImage
)
```

Parameters

Name	Type	Description
ipImage	ITcVnImage* [▶ 1925]&	Returns the undistorted image

Return value

HRESULT [▶ 150]

6.2.4.4.2.7 GetCurrentImageWithGvspInfo

Gets the current available image (first in receive queue) and its corresponding GVSP info.

Syntax

Definition:

```
HRESULT GetCurrentImageWithGvspInfo(
    ITcVnImage*&     ipImage,
    GVSP_IMAGE_INFO& stGvspInfo
)
```

Parameters

Name	Type	Description
ipImage	ITcVnImage* [▶ 1925]&	Returns the image
stGvspInfo	GVSP_IMAGE_INFO [▶ 1763]&	Returns the GVSP info

Return value

HRESULT [▶ 150]

6.2.4.4.2.8 GetCurrentImageWithGvspInfoUndistorted

Gets the current available image (first in receive queue) with its corresponding GVSP info and undistortion applied. Incompatible with pixel formats that contain an encoding, e.g. a Bayer pattern. Requires intrinsic calibration parameters in the TcVnGevImageProvider TcCOM module, e.g. set by the calibration assistant.

Syntax

Definition:

```
HRESULT GetCurrentImageWithGvspInfoUndistorted(
    ITcVnImage*&     ipImage,
    GVSP_IMAGE_INFO& stGvspInfo
)
```

Parameters

Name	Type	Description
ipImage	ITcVnImage* [▶ 1925]&	Returns the undistorted image
stGvspInfo	GVSP_IMAGE_INFO [▶ 1763]&	Returns the GVSP info

Return value

HRESULT [▶ 150]

6.2.4.4.2.9 GetCurrentImageWithTimestamps

Gets the current available image (first in receive queue) and corresponding timestamps.

Syntax

Definition:

```
HRESULT GetCurrentImageWithTimestamps(
    ITcVnImage*&    ipImage,
    ULONGLONG       nTimestamps,
    LONGLONG*       pTimestamps,
    GVSP_IMAGE_INFO& stGvspInfo
)
```

Parameters

Name	Type	Description
ipImage	ITcVnImage* [▶ 1925]&	Returns the image
nTimestamps	ULONGLONG	Select timestamps to return (ETcVnTimestamp)
pTimestamps	LONGLONG*	Returns the requested timestamps (make sure to provide an array of sufficient size).
stGvspInfo	GVSP_IMAGE_INFO [▶ 1763]&	Returns the GVSP info

Return value

HRESULT [▶ 150]

6.2.4.4.2.10 GetDistortionCoefficients

Gets the distortion coefficients from the TcVnGevImageProvider TcCOM module (requires intrinsic calibration parameters, e.g. set by the calibration assistant)

Syntax

Definition:

```
HRESULT GetDistortionCoefficients(
    TcVnArray8_LREAL& aDistortionCoefficients
)
```

Parameters

Name	Type	Description
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1713]&	Returns the distortion coefficients [k1, k2, p1, p2, k3, k4, k5, k6]

Return value

HRESULT [▶ 150]

6.2.4.4.2.11 GetLastImageFromQueue

Gets the last received image from the queue.

Syntax

Definition:

```
HRESULT GetLastImageFromQueue(
    ITcVnImage*& ipImage
)
```

Parameters

Name	Type	Description
ipImage	ITcVnImage* [▶ 1925]&	Returns the image

Return value

HRESULT [▶ 150]

6.2.4.4.2.12 GetOmittedImagesNum

Gets the number of omitted images since the last call of ClearImageQueue(). If the image receive queue in the TcVnGevImageProvider TcCOM module is full and a new image arrives, the first one in the queue will be deleted and the omitted counter is increased.

Syntax

Definition:

```
HRESULT GetOmittedImagesNum(
    ULONGLONG& nOmitted
)
```

Parameters

Name	Type	Description
nOmitted	ULONGLONG&	Returns the number of omitted images

Return value

HRESULT [▶ 150]

6.2.4.4.2.13 GetRotationMatrix

Gets the rotation matrix from the TcVnGevImageProvider TcCOM module (requires extrinsic calibration parameters, e.g. set by the calibration assistant)

Syntax

Definition:

```
HRESULT GetRotationMatrix(
    TcVnMatrix3x3_LREAL& aRotationMatrix
)
```

Parameters

Name	Type	Description
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	Returns the rotation matrix

Return value

HRESULT [▶ 150]

6.2.4.4.2.14 GetState

Gets the current state of the internal camera control state machine.

Syntax

Definition:

```
ETcVnCameraState GetState()
```

Return value

ETcVnCameraState [▶ 1718]

6.2.4.4.2.15 GetTranslationVector

Gets the translation vector from the TcVnGevImageProvider TcCOM module (requires extrinsic calibration parameters, e.g. set by the calibration assistant)

Syntax

Definition:

```
HRESULT GetTranslationVector(
    TcVnVector3_LREAL& aTranslationVector
)
```

Parameters

Name	Type	Description
aTranslationVector	TcVnVector3_LREAL [▶ 1713]&	Returns the translation vector

Return value

HRESULT [▶ 150]

6.2.4.4.2.16 InitializeCamera

Initialize the camera to the intended state (includes sending the 'InitCameraCommands' defined in the GevImageAcquisition TcCOM module to the camera).

Syntax

Definition:

```
HRESULT InitializeCamera()
```

Return value

HRESULT [▶ 150]

6.2.4.4.2.17 OpenCamera

Open a control channel to the camera.

Syntax

Definition:

```
HRESULT OpenCamera()
```

Return value

HRESULT [▶ 150]

6.2.4.4.2.18 Reset

Reset the camera controller to initial state (might require multiple calls depending on current state, until S_OK is returned)

Syntax

Definition:

```
HRESULT Reset()
```

Return value

HRESULT [▶ 150]

6.2.4.4.2.19 ResetCameraFeatures

Reset the camera features to initial state (sends the 'InitCameraCommands' defined in the GevImageAcquisition TcCOM module to the camera).

Syntax

Definition:

```
HRESULT ResetCameraFeatures()
```

Return value

HRESULT [▶ 150]

6.2.4.4.2.20 SetCameraMatrix

Sets the camera matrix to the TcVnGevImageProvider TcCOM module

Syntax

Definition:

```
HRESULT SetCameraMatrix(
    TcVnMatrix3x3_LREAL& aCameraMatrix
)
```

Parameters

Name	Type	Description
aCameraMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	The camera matrix to be copied to the TcCOM module

Return value

HRESULT [▶ 150]

6.2.4.4.2.21 SetDistortionCoefficients

Sets the distortion coefficients to the TcVnGevImageProvider TcCOM module

Syntax

Definition:

```
HRESULT SetDistortionCoefficients(
    TcVnArray8_LREAL& aDistortionCoefficients
)
```

Parameters

Name	Type	Description
aDistortionCoefficients	TcVnArray8_LREAL [▶ 1713]&	The distortion coefficients to be copied to the TcCOM module

Return value

HRESULT [▶ 150]

6.2.4.4.2.22 SetRotationMatrix

Sets the rotation matrix to the TcVnGevImageProvider TcCOM module

Syntax

Definition:

```
HRESULT SetRotationMatrix(
    TcVnMatrix3x3_LREAL& aRotationMatrix
)
```

Parameters

Name	Type	Description
aRotationMatrix	TcVnMatrix3x3_LREAL [▶ 1713]&	The rotation matrix to be copied to the TcCOM module

Return value

HRESULT [▶ 150]

6.2.4.4.2.23 SetTranslationVector

Sets the translation vector to the TcVnGevImageProvider TcCOM module

Syntax

Definition:

```
HRESULT SetTranslationVector(
    TcVnVector3_LREAL& aTranslationVector
)
```

Parameters

Name	Type	Description
aTranslationVector	TcVnVector3_LREAL [▶ 1713]&	The translation vector to be copied to the TcCOM module

Return value

HRESULT [▶ 150]

6.2.4.4.2.24 StartAcquisition

Send the 'StartAcquisitionCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.

Syntax

Definition:

```
HRESULT StartAcquisition()
```

Return value

HRESULT [▶ 150]

6.2.4.4.2.25 StopAcquisition

Send the 'StopAcquisitionCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.

Syntax

Definition:

```
HRESULT StopAcquisition()
```

Return value

HRESULT [▶ 150]

6.2.4.4.2.26 TriggerImage

Send the 'SoftwareTriggerCommands' defined in the TcVnGevImageProvider TcCOM module to the camera.

Syntax

Definition:

```
HRESULT TriggerImage()
```

Return value

HRESULT [▶ 150]

6.2.4.4.3 SimpleCameraControl

This FB provides the basic functionality to control a camera or a FileSource.

Syntax

Constructor:

```
SimpleCameraControl(
    OTCID oidITcVnImageProvider
)
```

Methods

Name	Description
GetCurrentImage [▶ 2777]	Get the current available image (if any).
GetState [▶ 2777]	Get the current state of the internal camera control state machine.
Reset [▶ 2778]	Reset the camera to initial state (might require multiple calls depending on current state, until S_OK is returned)
StartAcquisition [▶ 2778]	Start the image acquisition.
StopAcquisition [▶ 2778]	Stop the image acquisition.
TriggerImage [▶ 2778]	Trigger the next image.

Required License

TC3 Vision Base

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.4.3.1 GetCurrentImage

Get the current available image (if any).

Syntax

Definition:

```
HRESULT GetCurrentImage(
    ITcVnImage*& ipImage
)
```

Parameters

Name	Type	Description
ipImage	ITcVnImage* [▶ 1925]&	Returns the image

Return value

HRESULT [▶ 150]

6.2.4.4.3.2 GetState

Get the current state of the internal camera control state machine.

Syntax

Definition:

```
ETcVnCameraState GetState()
```

Return value

ETcVnCameraState [▶ 1718]

6.2.4.4.3.3 Reset

Reset the camera to initial state (might require multiple calls depending on current state, until S_OK is returned)

Syntax

Definition:

```
HRESULT Reset()
```

Return value

HRESULT [▶ 150]

6.2.4.4.3.4 StartAcquisition

Start the image acquisition.

Syntax

Definition:

```
HRESULT StartAcquisition()
```

Return value

HRESULT [▶ 150]

6.2.4.4.3.5 StopAcquisition

Stop the image acquisition.

Syntax

Definition:

```
HRESULT StopAcquisition()
```

Return value

HRESULT [▶ 150]

6.2.4.4.3.6 TriggerImage

Trigger the next image.

Syntax

Definition:

```
HRESULT TriggerImage()
```

Return value

HRESULT [▶ 150]

6.2.4.5 Image Processing

Diese Gruppe enthält Funktionsblöcke zur Bildverarbeitung. In diesen komplexen Funktionsblöcken bleiben die einmal gesetzten Werte nach einer Ausführung erhalten. Des Weiteren können die Funktionsblöcke auch mehrere Methoden enthalten.

Die im Kapitel Functions [▶ 1944] enthaltenen Funktionen haben hingegen keine internen Statusinformationen und die Variablenwerte bleiben nicht erhalten.

Funktionsblöcke

- GeneralizedHoughBallard [▶ 2779]
- SSIM [▶ 2782]

6.2.4.5.1 GeneralizedHoughBallard

This FB provides the Generalized Hough Ballard functionality. First, set the parameters via the corresponding methods (optional, defaults are used otherwise). Then, set the template (required). Afterwards, the Detect method can be executed.

Syntax

Constructor:

```
GeneralizedHoughBallard()
```

Methods

Name	Description
Detect [▶ 2779]	Detect the template in an image (SetTemplate must have been called before).
SetCannyThreshold [▶ 2780]	Sets the canny edge detection thresholds.
SetInvAccuRatio [▶ 2780]	Sets the inverted ratio of the accumulator size in relation to the source image's size (e.g. a value of 2 means that the size is halved in both directions.)
SetLevels [▶ 2781]	Sets the number of R-table levels.
SetMinDist [▶ 2781]	Sets the minimum distance between the centers of different objects.
SetTemplate [▶ 2781]	Sets the template image to search for in the Detect method.
SetVotesThreshold [▶ 2782]	Sets the accumulator threshold, i.e. the number of votes required to detect a match (too small values lead to false detections).

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.5.1.1 Detect

Detect the template in an image (SetTemplate must have been called before).

Syntax

Definition:

```
HRESULT Detect(
    ITcVnImage*      ipImage,
    ITcVnContainer*& ipPositions
)
```

Parameters

Name	Type	Description
ipImage	ITcVnImage* [▶ 1925]	Source image (1 channel, USINT)
ipPositions	ITcVnContainer* [▶ 1890]&	Returns the centers of the detected template positions (ContainerType_Vector_TcVnPoint2_REAL; Non-zero interface pointers are reused.)

Return value

HRESULT [▶ 150]

6.2.4.5.1.2 SetCannyThreshold

Sets the canny edge detection thresholds.

Syntax

Definition:

```
HRESULT SetCannyThreshold(
    LONG nLow = 50,
    LONG nHigh = 100
)
```

Parameters

Name	Type	Description
nLow	LONG	Low threshold (> 0)
nHigh	LONG	High threshold (> nLow, usually 2 to 3 * nLow)

Return value

HRESULT [▶ 150]

6.2.4.5.1.3 SetInvAccuRatio

Sets the inverted ratio of the accumulator size in relation to the source image's size (e.g. a value of 2 means that the size is halved in both directions.)

Syntax

Definition:

```
HRESULT SetInvAccuRatio(
    double fInvAccuRatio = 1.0
)
```

Parameters

Name	Type	Description
fInvAccuRatio	double	Inverted accumulator ratio (> 0)

Return value

HRESULT [▶ 150]

6.2.4.5.1.4 SetLevels

Sets the number of R-table levels.

Syntax

Definition:

```
HRESULT SetLevels(
    LONG nLevels = 360
)
```

Parameters

Name	Type	Description
nLevels	LONG	Number of R-table levels (> 0)

Return value

HRESULT [▶ 150]

6.2.4.5.1.5 SetMinDist

Sets the minimum distance between the centers of different objects.

Syntax

Definition:

```
HRESULT SetMinDist(
    double fMinDist = 1.0
)
```

Parameters

Name	Type	Description
fMinDist	double	Minimum distance (> 0)

Return value

HRESULT [▶ 150]

6.2.4.5.1.6 SetTemplate

Sets the template image to search for in the Detect method.

Syntax

Definition:

```
HRESULT SetTemplate(
    ITcVnImage* ipImage
)
```

Parameters

Name	Type	Description
ipImage	ITcVnImage* [▶ 1925]	Template image (1 channel, USINT)

Return value

HRESULT [▶ 150]

6.2.4.5.1.7 SetVotesThreshold

Sets the accumulator threshold, i.e. the number of votes required to detect a match (too small values lead to false detections).

Syntax

Definition:

```
HRESULT SetVotesThreshold(
    LONG nVotes = 100
)
```

Parameters

Name	Type	Description
nVotes	LONG	Number of votes required to detect a match (> 0)

Return value

HRESULT [▶ 150]

6.2.4.5.2 SSIM

This FB provides the structural similarity (SSIM) functionality. First, set the reference image. Afterwards, the Compute method can be executed.

Syntax

Constructor:

```
SSIM()
```

Methods

Name	Description
Compute [▶ 2783]	Compute the structural similarity (SetReference must have been called before).
SetReference [▶ 2783]	Sets the reference image, used by the SSIM method.

Required License

TC3 Vision Matching

System Requirements

Development environment	Target platform	PLC libraries to include
TwinCAT V3.1.4024.59 or later	PC or CX (x64) with min. PL50, e.g. Intel 4-core Atom CPU	Tc3_Vision

6.2.4.5.2.1 Compute

Compute the structural similarity (SetReference must have been called before).

Syntax

Definition:

```
HRESULT Compute(
    ITcVnImage*  ipSrcImage,
    ITcVnImage*& ipSSIM
)
```

Parameters

Name	Type	Description
ipSrcImage	ITcVnImage* [▶ 1925]	Source image (USINT)
ipSSIM	ITcVnImage* [▶ 1925]&	Returns the SSIM (REAL, values from 0 to 1)

Return value

HRESULT [▶ 150]

6.2.4.5.2.2 SetReference

Sets the reference image, used by the SSIM method.

Syntax

Definition:

```
HRESULT SetReference(
    ITcVnImage* ipImage
)
```

Parameters

Name	Type	Description
ipImage	ITcVnImage* [▶ 1925]	Reference image (USINT)

Return value

HRESULT [▶ 150]

7 TwinCAT HMI Pakete

Dieses Kapitel behandelt das Zusammenspiel zwischen TwinCAT Vision und der TwinCAT HMI. Dies besteht aus einer Server-Erweiterung [▶ 2784], mit der Bilder [▶ 159] aus der SPS in der TwinCAT HMI angezeigt werden können sowie einem Controls-Paket [▶ 2789] zur Visualisierung und Interaktion mit Bilder und Farben.

Einbinden von Paketen

Das Einbinden von Paketen erfolgt bei TwinCAT HMI Versionen ab 1.12 über die Paketverwaltung NuGet. Machen Sie in einem TwinCAT HMI-Projekt einen Rechtsklick auf **References** und wählen Sie **Manage NuGet Packages…**. Klicken Sie dann auf **Browse** und wählen als **Package source** „TwinCAT HMI Official“ oder „Nuget.org“ aus. Sie sehen nun eine Liste der verfügbaren TwinCAT HMI Pakete, aus der Sie z.B. das **Beckhoff.TwinCAT.HMI.Vision** oder das **Beckhoff.TwinCAT.HMI.Vision.Controls** auswählen können.

Ab der TwinCAT HMI Version 1.14 können die Pakete **TwinCAT.HMI.Vision** und **TwinCAT.HMI.Vision.Controls** über den TwinCAT Package Manager installiert werden. Danach müssen die Pakete, wie zuvor beschrieben, über die **Package source** "Beckhoff Offline Packages" in das HMI-Projekt eingebunden werden.

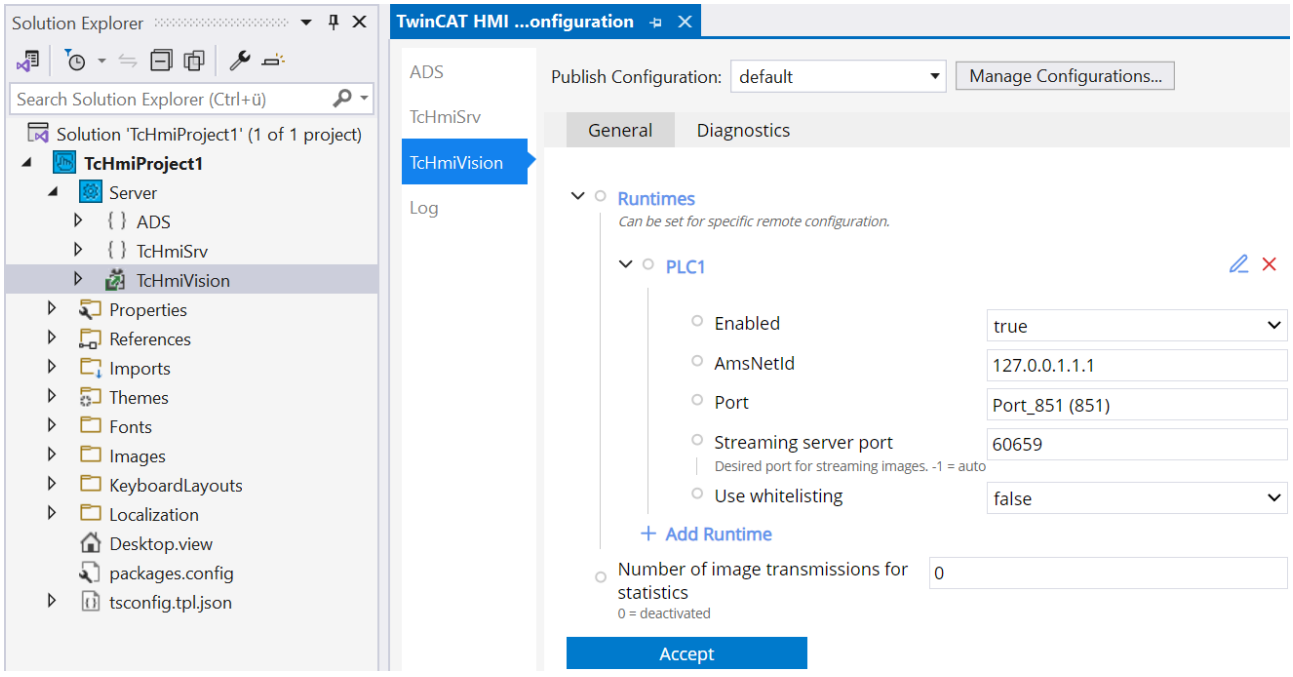
Generelle Informationen entnehmen Sie der TwinCAT HMI Dokumentation z.B. mit der detaillierten Beschreibung zur Paketverwaltung.

7.1 Server-Erweiterung

Das NuGet-Paket Beckhoff.TwinCAT.HMI.Vision (TwinCAT-Paket „TwinCAT.HMI.Vision“) enthält die HMI Server-Erweiterung `TcHmiVision`, mit der Bilder aus der SPS in der TwinCAT HMI angezeigt werden können. Dazu werden alle ADS-Symbole vom Typ ITcVnDisplayableImage [▶ 426] aus den konfigurierten Runtimes aufgelistet und können in verschiedenen Formaten mit einem HMI-Symbol verknüpft werden. Siehe Systemvoraussetzungen [▶ 15] für die benötigte .NET Version.

Konfiguration

Nachdem das Paket erfolgreich in das HMI-Projekt eingebunden wurde, tragen Sie unter **Server > TcHmiVision > General** die Ziel-SPS ein:



Zudem können einzelne Zielsysteme über die Einstellung **Enabled** aktiviert oder deaktiviert werden.

Der **Streaming server port** definiert, welcher Streaming Port für das Senden der Bilder, die mit `Stream` als Übertragungsformat konfiguriert wurden, verwendet wird.

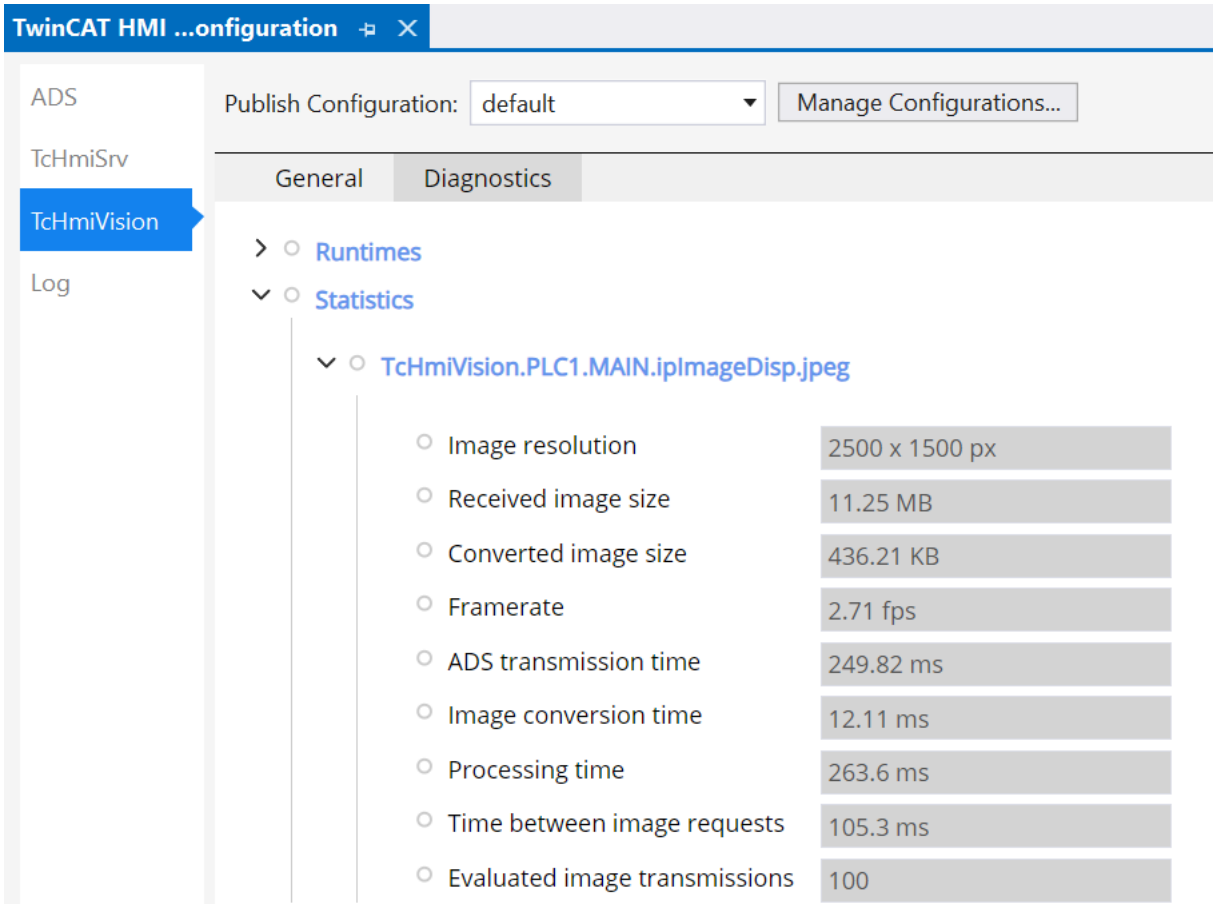
Wenn einzelne Bildsymbole aus der SPS von der Extension nicht gelistet werden sollen, können diese mit dem SPS-Attribut `TcHmiSymbol.Hide` ausgeblendet werden. Wenn dagegen nur bestimmte Bildsymbole gelistet werden sollen, muss die Einstellung **Use Whitelisting** aktiviert werden. Dann können entsprechende Variablen mit dem SPS-Attribut `TcHmiSymbol.Show` gekennzeichnet werden.

```
VAR
    { attribute 'TcHmiSymbol.Hide' } // or: TcHmiSymbol.Show
    ipDisplayableImage: ITcVnDisplayableImage;
END_VAR
```

Statistiken

Setzen Sie unter **Server > TcHmiVision > General** die `Number of image transmissions for statistics` auf einen höheren Wert als 0, um die Erfassung der statistischen Daten zu aktivieren. Wenn Sie beispielsweise den Wert 100 einstellen, fließen die letzten 100 Bildübertragungen in die Statistik ein.

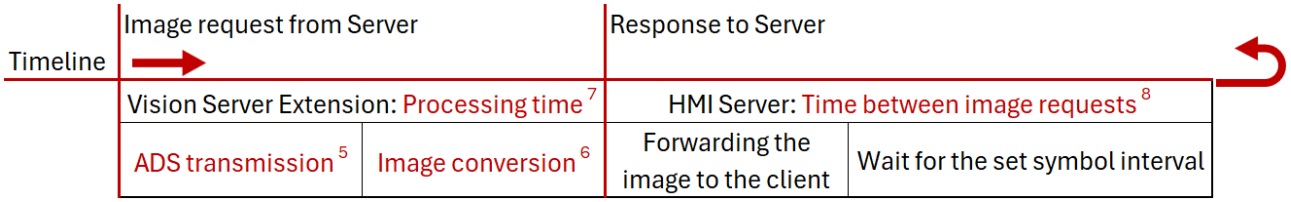
Unter **Server > TcHmiVision > Diagnostics > Statistics** werden statistische Daten aufgelistet. Die einzelnen Werte werden pro konfiguriertem Bild-Symbol berechnet.



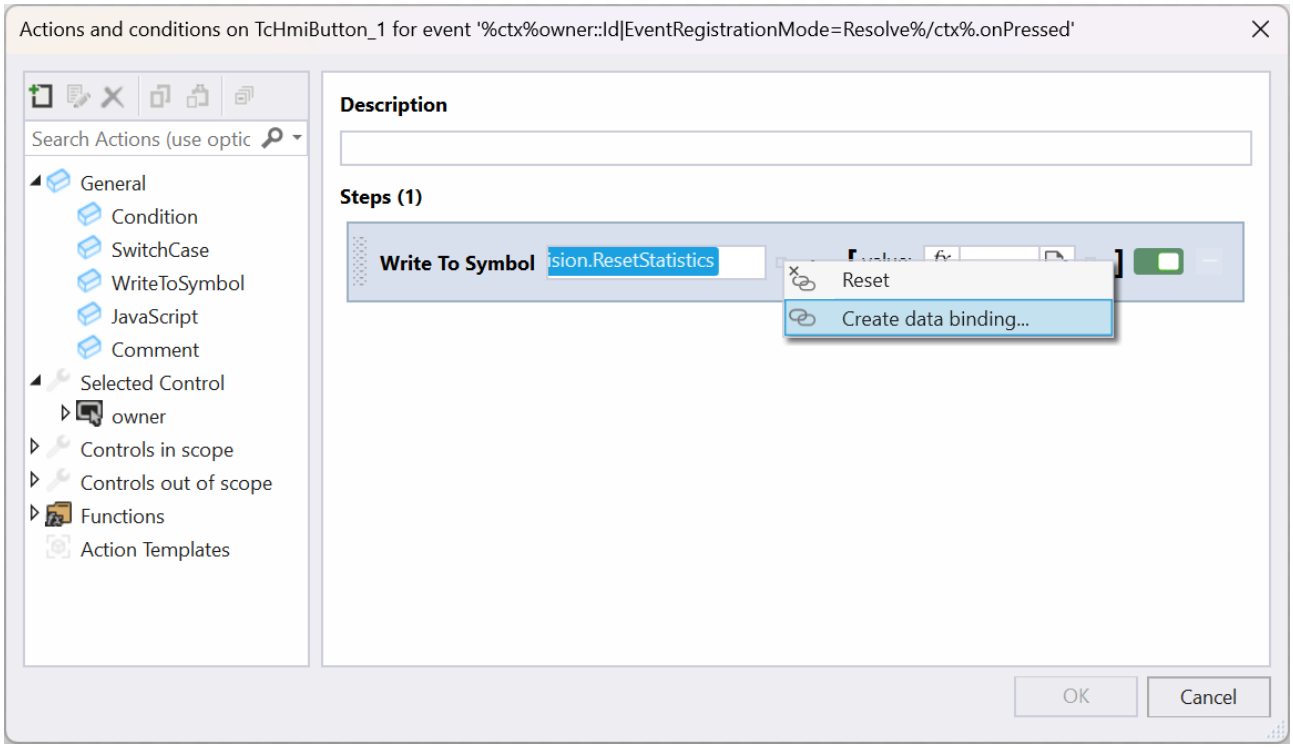
1. Image resolution: Bildauflösung in Pixeln
2. Received image size: Bildgröße des empfangenen Bildes
3. Converted image size: Bildgröße nach der Konvertierung zuerst z.B. nach Jpeg oder Png abhängig vom konfigurierten Bildformat und dann in einen Base64-String
4. Framerate: Bildübertragungen pro Sekunde
5. ADS transmission time: Dauer der Übertragung eines Bildes per ADS
6. Image conversion time: Dauer der Konvertierung eines Bildes
7. Processing time: Die Zeit, die von der Extension für eine Bildübertragung inklusive Konvertierung in Anspruch genommen wird.

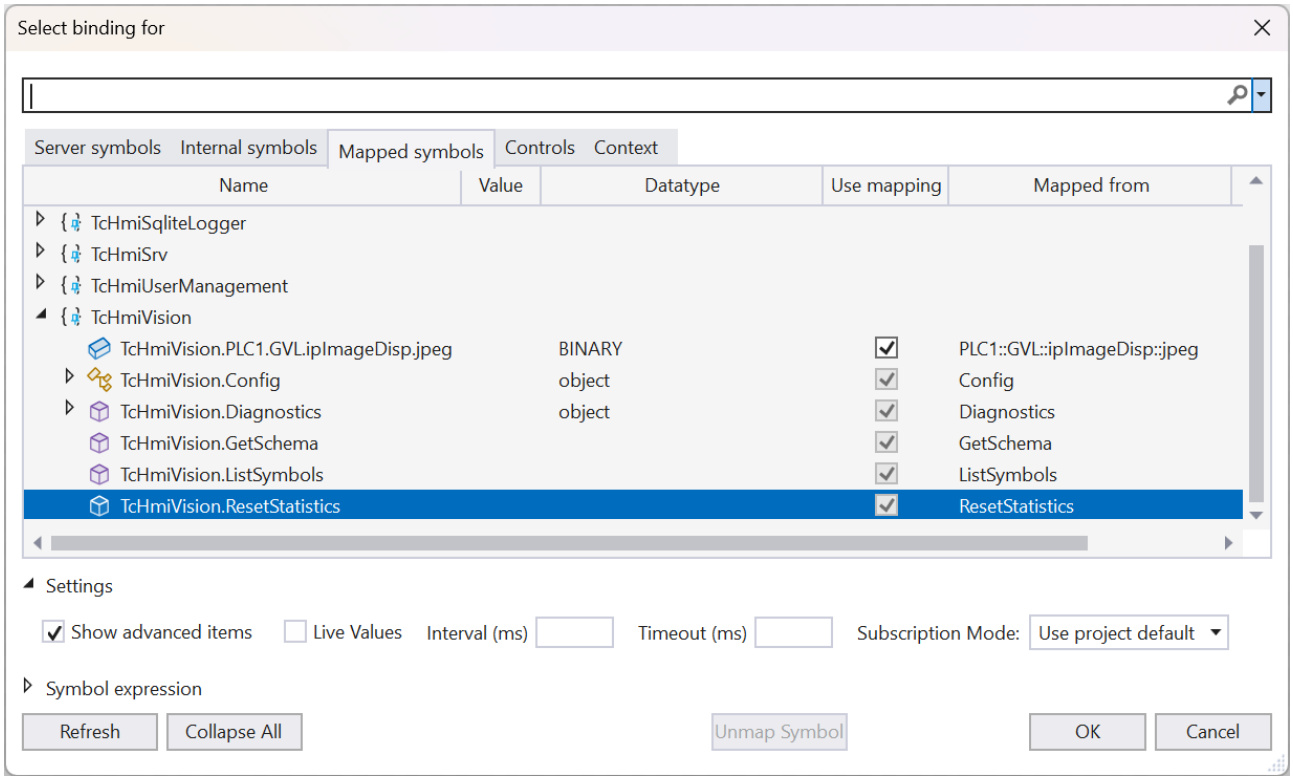
8. Time between image requests: Die Zeit, die von einer Antwort der Extension bis zur nächsten Bildanfrage des Servers vergeht. Das eingestellte Symbolintervall wird in dieser Phase vom Server abgewartet. Wenn die Bildübertragung pausiert wurde, kann dieser Wert verzerrt sein. Um verfälschte Werte zu verhindern, können die Statistiken wie nachfolgend erklärt, zurückgesetzt werden. Alternativ kann auch gewartet werden, bis weitere Bilder übertragen wurden und die Statistik angepasst wurde. Weiterhin kann dieser Wert variieren, wenn mehrere Databindings auf dasselbe Symbol existieren.
9. Evaluated image transmissions: Anzahl der Bildübertragungen, die aktuell in die Statistik einfließen

Grafische Darstellung des Ablaufs einer Bildübertragung, mit den in rot markiert statistischen Werten:



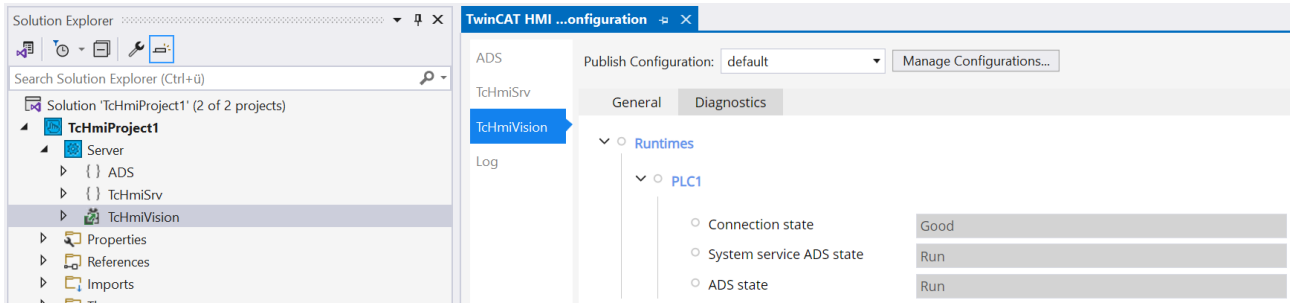
Um die Statistiken aus der Applikation zurückzusetzen, muss ein leerer Wert in das Symbol `TcHmiVision.ResetStatistics` geschrieben werden. Ziehen Sie dafür beispielsweise ein **Write To Symbol** in die Actions and Conditions eines Events. Klicken Sie beim linken Feld auf **Create data binding…** Damit das Symbol aufgelistet wird, müssen Sie in den Einstellungen des Fensters **Show advanced items** aktivieren. Verlinken Sie jetzt das entsprechende Symbol unter: **Mapped symbols > TcHmiVision > ResetStatistics**.





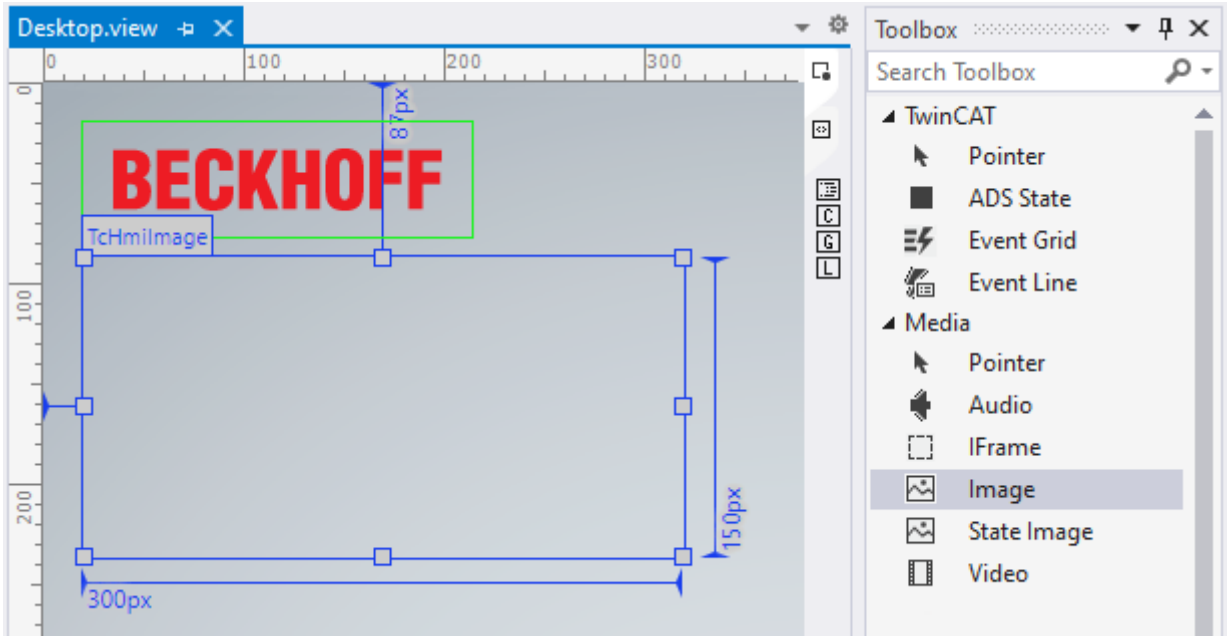
Diagnose

Unter **Server > TcHmiVision > Diagnostics** erhalten Sie Informationen zum Verbindungsstatus des konfigurierten Zielsystems:

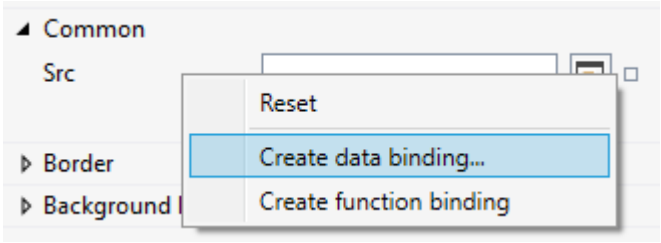


Anwendung

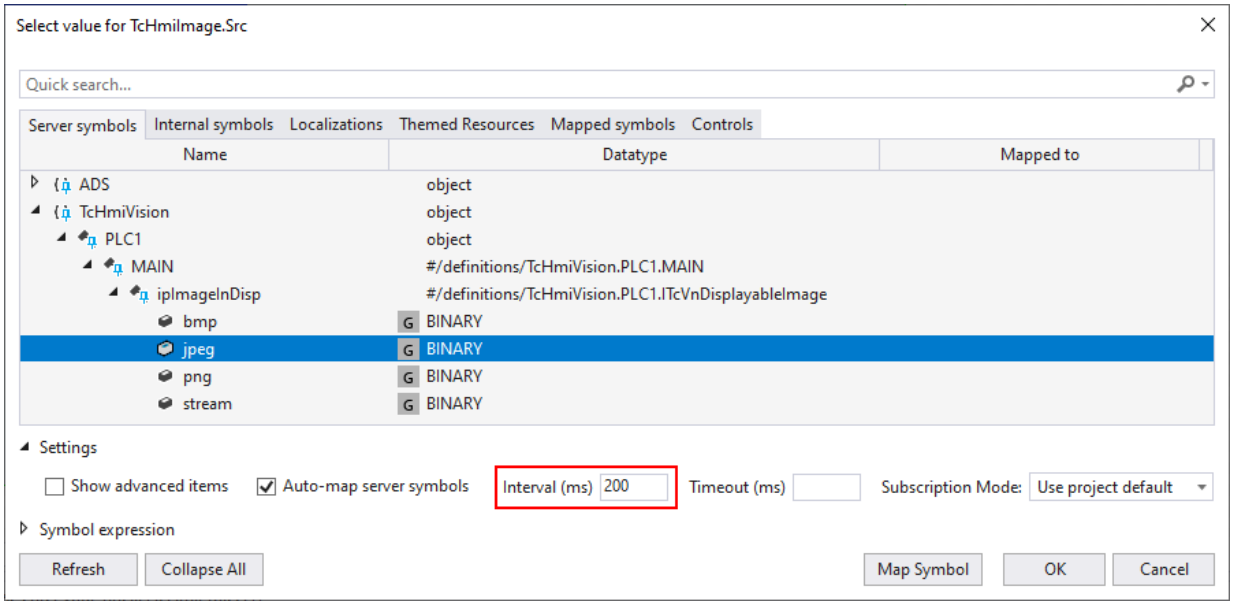
1. Navigieren Sie zu einem HMI-View und fügen Sie ein **Image**-Control ein.



2. Öffnen Sie die Eigenschaften des Controls und klicken Sie beim Parameter **Src** auf **Create data binding…**



3. Verlinken Sie den Src-Parameter des Image-Controls mit dem entsprechenden Bildsymbol an folgender Stelle: **Server symbols > TcHmiVision > PLC > MAIN > ipImage > <Bildformat>**



Verlinken von Bildern

Das Verlinken von Bildern ist nur möglich, wenn im Prozessabbild des konfigurierten Zielsystems entsprechende Bildvariablen enthalten sind.

Intervall

Die Geschwindigkeit, mit der Bilder aus der SPS abgerufen werden können, hängt von dem System, der Bildgröße und der Auslastung ab. Wir empfehlen daher, im Binding-Dialog unter **Settings** ein festes Intervall anzugeben um eine Überlastung des Systems zu verhindern.

Wenn mehrere Bilder auf einer Seite angezeigt werden sollen, empfiehlt es sich weiterhin, unterschiedliche Intervalle zu verwenden z. B. 200 ms und 201 ms. Dadurch werden die Bilder getrennt voneinander abgerufen, was bei größeren Datenmengen vorteilhaft sein kann.

Übertragungsformate

Die Server-Erweiterung erhält die Bilddaten von der SPS im BMP-Format und sendet sie als Base64-String an den TwinCAT HMI Client weiter. Um die Datenmenge, die an einen Client übertragenen wird zu reduzieren, können Bilder in der Server-Erweiterung komprimiert werden. Folgende Bildformate stehen zur Verfügung:

- **BMP**: unkomprimiert
- **JPEG**: verlustbehaftet komprimiert, gut für natürliche Bilder
- **PNG**: verlustfrei komprimiert, gut für künstlich erstellte Bilder
- **Stream**: Für kontinuierliche Bild-Streams (z. B. Live-Bild der Kamera, nicht empfohlen!)

Für die Übertragung als Stream muss der HMI-Server als Administrator gestartet werden. Sonst ist keine Bildübertragung möglich.

7.2 Controls

Das NuGet-Paket Beckhoff.TwinCAT.HMI.Vision.Controls (TwinCAT-Paket „TwinCAT.HMI.Vision.Controls“) stellt einige Controls zur Anzeige und Manipulation von Vision-Daten bereit. Diese Dokumentation setzt grundlegende Kenntnis von TwinCAT HMI Konzepten voraus siehe TE2000 und bezieht sich auf diese. Eine detaillierte Beschreibung zur Nutzung von Controls ist ebenfalls dort zu finden.

Controls

Das Paket enthält folgende Controls, die über die HMI-Toolbox zu HMI-Seiten hinzugefügt werden können:

Control	Kategorie	Beschreibung
Image Watch [▶ 2790]	Vision	• Anzeige von Bildern • Zoom- und Drag-Optionen • Extrahieren von Bildinformationen • Markieren von Bildbereichen mit verschiedenen Shapes • Multi-Bild-Auswahl • Overlay-Grafiken • Toolbar mit Bedienfunktionen • Infobar zur Anzeige von Bild- und Shape-Informationen
Color [▶ 2815]	Vision	Anzeige und Einstellen von Farben
Rectangle (Interactive) [▶ 2828]	Shapes	Interaktives Anzeigen und Zeichnen eines Rechtecks.
Polygon (Interactive) [▶ 2830]	Shapes	Interaktives Anzeigen und Zeichnen eines Polygons.

Form-Controls

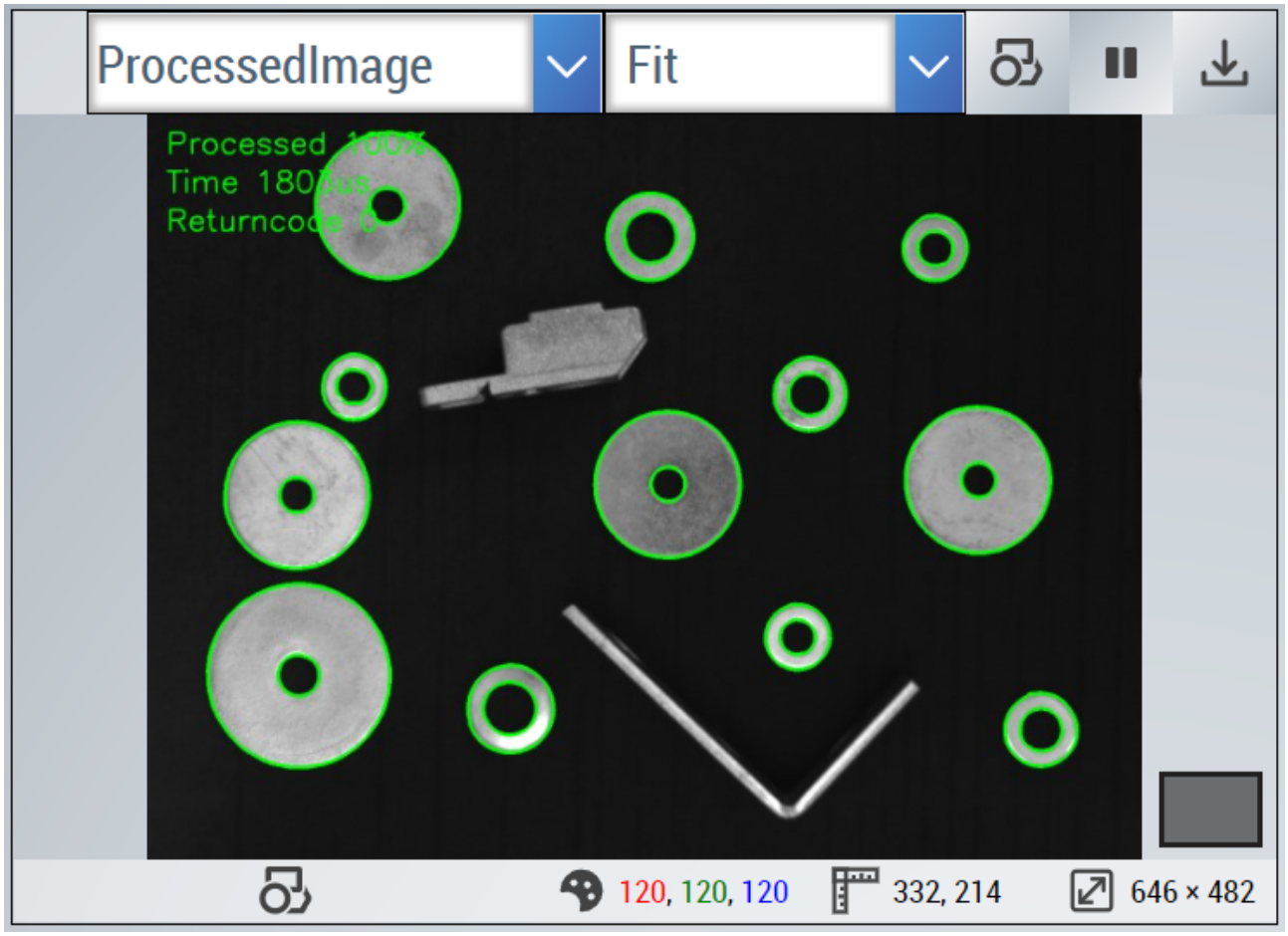
Die Controls Rectangle (Interactive) und Polygon (Interactive) sind primär Hilfs-Controls, die im Image Watch Control wiederverwendet werden. Bei Bedarf können sie auch alleinstehend verwendet werden. Zudem können die Controls mit Hilfe ihrer Attribute auch für abgeleitete Formen wie einen Punkt, eine Linie, einen Kreis, eine Ellipse oder ein Quadrat benutzt werden.

Zusätzlich enthält das Paket folgende Hilfsfunktionen:

Funktion	Kategorie	Beschreibung
BoxColorConverter [▶ 2832]	Vision	Konvertiert das `ColorValue` Attribut des Color [▶ 2815] Controls für die Anzeige in der Farbbox und umgekehrt.
ConvertColor [▶ 2833]	Vision	Konvertiert Farbwerte zwischen verschiedenen Formaten.
PixelColorFormatter [▶ 2834]	Formatting	Formatiert das `PixelColor` Attribut des Image Watch [▶ 2790] Controls in einen Text zur Anzeige in der Toolbar [▶ 2804].
ShapeValueFormatter [▶ 2834]	Formatting	Formatiert das `ShapeValue` Attribut des Image Watch [▶ 2790] Controls in einen Text zur Anzeige in der Toolbar [▶ 2804].
ToRotatedRectangle [▶ 2834]	Data Conversion	Konvertiert ein Rechteck vom Typ TcVnRectangle [▶ 2836] zum Typ TcVnRotatedRectangle [▶ 2836].
ToUprightRectangle [▶ 2835]	Data Conversion	Konvertiert ein Rechteck vom Typ TcVnRotatedRectangle [▶ 2836] zum Typ TcVnRectangle [▶ 2836].

7.2.1 Image Watch

Das **Image Watch** Control bietet zusätzlich zu der Bildanzeige noch viele weitere Werkzeuge und Informationsanzeigen. Eine Übersicht über die Bereiche und Komponenten finden Sie im Unterkapitel Aufbau des Controls [▶ 2798]. Die Beschreibung der einzelnen Konfigurationsmöglichkeiten finden Sie unter den folgenden Kategorien auf dieser Seite. Zu den jeweiligen Kategorien gibt es Unterkapitel mit weiteren Informationen, Details und Anwendungsbeispielen. Um direkt TwinCAT Vision Bilder aus der SPS anzeigen zu können, wird zudem die Vision-Erweiterung [▶ 2784] für den HMI-Server benötigt.



Für alle folgenden Attribute besitzt das Control eine Getter- und eine Setter-Methode. Zum Beispiel existieren für das Attribut `BarColor` die beiden Methoden `getBarColor(): SolidColor` und `setBarColor(value: SolidColor): void`. Die Benutzung der Attribute wird in den verlinkten Unterkapiteln näher erklärt.

Alle weiteren Attribute, Events und Permissions die die Basisklasse **Controls** bereitstellt, finden Sie in der TwinCAT HMI Dokumentation.

Kategorie: Colors

Attribut	Typ	Beschreibung
`BarColor`	`SolidColor`	Farbe der Toolbar und der Infobar (siehe Kategorien **Toolbar** und **Infobar**).
`ShapeStrokeColor`	`SolidColor`	Farbe der Konturen der Formen, die auf dem Bild gezeichnet werden können (siehe Kategorie **Shapes**).
`ShapeHandleColor`	`SolidColor`	Farbe der Griffe der Formen, die auf dem Bild gezeichnet werden können (siehe Kategorie **Shapes**).

Kategorie: Common

Die folgenden Attribute legen die Ressource [▶ 2799] fest, von der die anzuzeigenden Bilder geladen werden.

Attribut	Typ	Beschreibung
`Image`	`String`	Ressource für das anzuzeigende Bild. Kann entweder ein Pfad zu einer Bilddatei oder eine Daten-URI sein.
`ImageList`	`ImageList`	Liste mit benannten Bild-Ressourcen. Bei nicht-leerer Liste wird das Binding von `Image` mit der durch `ImageIndex` ausgewählten Ressource ersetzt.
`ImageIndex`	`Number`	Wählt bei nicht-leerer `ImageList` aus, welche der Bildressourcen aus der Liste angezeigt wird.
`Alt`	`String`	Alternativ-Text, der angezeigt wird, wenn die aktuelle Bild-Ressource nicht verfügbar ist.

Funktion	Beschreibung
`downloadImage(fileName: string): void`	Löst einen browserspezifischen Download des aktuell angezeigten Bildes in Originalgröße aus. Kann alternativ über ein UI-Element in der Toolbar [▶ 2804] ausgelöst werden.

Kategorie: View

Die folgenden Attribute und Funktionen bestimmen den Ausschnitt des angezeigten Bildes [▶ 2801] und wie dieser verändert werden kann.

Attribute	Typ	Beschreibung
ImageFreeze	Boolean	Wenn aktiv, ist das aktuell angezeigte Bild festgesetzt und wird nicht mehr aktualisiert bis ImageFreeze deaktiviert wird.
Offset	[Number, Number]	Position der oberen linken Bild-Ecke im Koordinatensystem des Controls.
Scale	Number (%)	Skalierung des Bildes in Prozent basierend auf ScaleReference.
ScaleReference	ScaleReference	Bei ImageSize entsprechen 100% der originalen Bildgröße. Bei ControlWidth entsprechen 100% der Breite des Controls. Bei ControlHeight entsprechen 100% der Höhe des Controls. Bei ControlSize entsprechen 100% entweder der Breite oder der Höhe des Controls, je nach dem welches die Größe des Bildes zuerst beschränkt.
Draggable	Boolean	Wenn aktiv, kann das Bild mittels Maus oder Touch verschoben werden. Dabei wird Offset automatisch angepasst.
Scalable	Boolean	Wenn aktiv, kann das Bild mittels Mausrad oder Multitouch skaliert werden. Dabei werden Offset und Scale automatisch angepasst.
ScaleMin	Number	Minimum Scale bei der Skalierung mittels Mausrad oder Multitouch.
ScaleMax	Number	Maximum Scale bei der Skalierung mittels Mausrad oder Multitouch.
ScalingCenter	ScalingCenter	Punkt, der bei Skalierung über das Scale Attribute fixiert bleibt. Bei TopLeftCorner bleibt die obere linke Ecke des Bildes fixiert. Bei ImageCenter bleibt der Mittellpunkt des Bildes fixiert. Bei ControlCenter bleibt der Mittelpunkt des Controls fixiert. Bei Skalierung mittels Mausrad bleibt immer die Position des Cursors fixiert.
View (readonly)	Number[4]	Gibt den angezeigten Ausschnitt des Bildes im Sinne einer Region of Interest. Das Format ist: [X, Y, Breite, Höhe].

Funktion	Beschreibung
`centerImageToControl(): void`	Passt das `Offset` derart an, dass das aktuelle Bild mittig im Control ist.
`fitImageToControl(center: boolean): void`	Passt `Offset` und `Scale` derart an, dass das aktuelle Bild mittig und ausgefüllt im Control ist.
`setView(view: number[]): void`	Verändert `Offset` und `Scale`, sodass der spezifizierte Ausschnitt des Bildes angezeigt wird. Muss nicht zwangsläufig mit View übereinstimmen, wenn das Seitenverhältnis des Ausschnitts nicht mit dem Control übereinstimmt.

Kategorie: Toolbar

Mit folgenden Attributen kann die Toolbar [▶ 2804] des Controls angepasst werden.

Attribut	Typ	Beschreibung
`ToolbarVisible`	`Boolean`	Anzeige der Toolbar.
`ToolbarItems`	`ToolbarItems`	Auswahl von Elementen, die in der Toolbar angezeigt werden.
`ScaleOptions`	`Number[]`	Skalierungswerte, die in der `ScaleSelection` Combobox zusätzlich zur „Fit“-Option angezeigt werden und auswählbar sind.
`ToolbarHeight`	`Number (px)`	Höhe der Toolbar.
`ImageSelectionWidth`	`Number (px)`	Breite der `ImageSelection` Combobox.
`ScaleSelectionWidth`	`Number (px)`	Breite der `ScaleSelection` Combobox.
`ScaleSelectionEditable`	`Boolean`	Wenn aktiv, können manuell Werte in die `ScaleSelection` eingetippt werden. Wenn inaktiv, kann nur aus den angebotenen `ScaleOptions` ausgewählt werden.

Kategorie: Infobar

Mit folgenden Attributen kann die Infobar [▶ 2805] des Controls angepasst werden.

Attribut	Typ	Beschreibung
`InfobarVisible`	`Boolean`	Anzeige der Infobar.
`InfobarItems`	`InfobarItems`	Auswahl von Elementen, die in der Infobar angezeigt werden.
`PixelColorFormat`	`Function`	Funktion, die aus dem Wert des `PixelColor`-Attributs einen Text für die Infobar generiert.
`ShapeValueFormat`	`Function`	Funktion, die aus dem Wert des `ShapeValue`-Attributs einen Text für die Infobar generiert.
`InfobarHeight`	`Number (px)`	Höhe der Infobar.

Kategorie: Shapes

Folgende Attribute helfen beim Zeichnen von Formen [▶ 2808] auf dem Bild.

Attribut	Typ	Beschreibung
ShapeSelectionActive	Boolean	Wenn aktiv, kann auf dem Bild eine Form (basierend auf ShapeType) gezeichnet, angezeigt und bearbeitet werden.
ShapeType	ShapeType	Typ der Form. Möglich sind: • Punkt • Linie • Rechteck / Quadrat • Ellipse / Kreis • Polygon
ShapeIsRotatable	Boolean	Wenn aktiv, können geeignete Formen mittels eines Handles in der UI rotiert werden. (Linie, Rechteck, Quadrat, Ellipse, Polygon)
ShapeData	ShapeData	Geometrische Beschreibung der Form. Kann genutzt werden, um die Daten einer gezeichneten Form abzufragen oder um eine bestimmte, vordefinierte Form anzuzeigen.
ShapeAutoClear	Boolean	Wenn aktiv, wird ShapeData auf den Basiswert des jeweiligen ShapeType zurückgesetzt, wenn ShapeSelectionActive deaktiviert wird.
ShapeStrokeThickness	Number (px)	Dicke der Linien der angezeigten Form.
ShapeHandleSize	Number (px)	Größe der Handles der angezeigten Form.
ShapeClickableSize	Number (px)	Größe der klickbaren Fläche für die Handles der angezeigten Form. Kann genutzt werden, um die Bearbeitung der Form auf Touch-Panels zu vereinfachen, ohne die Handles unnötig groß anzuzeigen.
ShapeAngleInterval	Number (°)	Schrittgröße, in der die Rotation der Form geändert werden kann. Z. B. gibt es bei einem Wert von 90 nur vier mögliche Ausrichtungen. Bei 0 ist die Rotation beliebig.
ShapeIsInImageArea (readonly)	Boolean	Gibt an, ob die angezeigte Form vollständig innerhalb des Bildbereichs liegt oder nicht. Kann hilfreich sein, um die Validität von ShapeData für bestimmte PLC-Funktionen zu überprüfen. Dabei werden nur die Eckpunkte zur Berechnung verwendet.
ShapeLimitToImage	Boolean	Wenn aktiv, kann die Form nur innerhalb des Bildes gezeichnet oder bearbeitet werden.
ShapeDataUpdate	DataUpdate	Legt fest, ob ShapeData während (onChanged) oder am Ende (onConfirmed) einer grafischen Interaktion aktualisiert werden soll.

Funktion	Beschreibung
`clearShape(): void`	Löscht die angezeigte Form, wobei auch `ShapeData` auf den Basiswert des jeweiligen `ShapeType` zurückgesetzt wird.

Kategorie: Thumbnail

Folgende Attribute legen die Anzeige eines Thumbnails fest. Der Thumbnail [▶ 2812] gibt einen Überblick, welcher Ausschnitt des Bildes sichtbar ist.

Attribut	Typ	Beschreibung
`ThumbnailVisible`	`Boolean`	Anzeige des Thumbnails.
`ThumbnailPosition`	`ThumbnailPosition`	Position des Thumbnails.
`ThumbnailSize`	`Number (px)`	Größe des Thumbnails. Bestimmt die Breite des Thumbnails; die Höhe wird automatisch anhand des Seitenverhältnisses berechnet.

Kategorie: Overlay

Die folgenden Attribute ermöglichen das Einblenden von Objekten [▶ 2813] auf dem Bild (z. B. als Positionierungs- oder Markierungshilfen).

Attribut	Typ	Beschreibung
`OverlayVisible`	`Boolean`	Anzeige von Einblendungen.
`OverlayImage`	`String`	Bild, das über dem angezeigten Bild eingeblendet wird.
`OverlayElements`	`OverlayElements`	Geometrische Elemente, die über dem angezeigten Bild eingeblendet werden.

Kategorie: Info

Folgende Attribute können genutzt werden, um Informationen [▶ 2814] über das Bild oder bestimmte Pixel des Bildes zu erlangen.

Attribut	Typ	Beschreibung
`PixelInfoUpdate`	`PixelInfoUpdate`	Bestimmt, wann `PixelColor` und `PixelPosition` aktualisiert werden. • Bei `Never` werden sie nie aktualisiert. • Bei `AtClick` werden sie bei einem Klick auf eine bestimmte Stelle des Bildes aktualisiert. • Bei `AtCursor` werden sie bei jeder Bewegung des Cursors auf dem Bild aktualisiert. Damit werden immer die Werte der aktuellen Cursor-Position angezeigt.
`PixelColor` (readonly)	`[Number, Number, Number, Number]`	Farbe des ausgewählten Bild-Pixels.
`PixelPosition` (readonly)	`[Number, Number]`	Position des ausgewählten Bild-Pixels in Bildkoordinaten.
`ImageSize` (readonly)	`[Number, Number]`	Original-Größe des angezeigten Bildes.
`ImageDisplaySize` (readonly)	`[Number, Number]`	Darstellungsgröße des angezeigten Bildes.

Events

Alle Events sind in der Kategorie „Vision“ gelistet.

Event	Beschreibung
`.onPixelInfo`	Dieses Event wird ausgelöst, wenn `PixelInfoUpdate <> Never` ist und der Nutzer auf eine Position im Bild klickt oder den Mauszeiger darüber bewegt und so ein Pixel auswählt. Intern werden mit dem Event die Werte der Attribute `PixelColor` und `PixelPosition` aktualisiert.
`.onImageReceived`	Dieses Event wird ausgelöst, wenn das Control ein Bild empfängt.
`.onShapeChanged`	Dieses Event wird ausgelöst, sobald sich das Attribut `ShapeData` ändert.
`.onShapeConfirmed`	Dieses Event wird ausgelöst, wenn eine Form gesetzt oder eine Interaktion mit dieser abgeschlossen wurde.

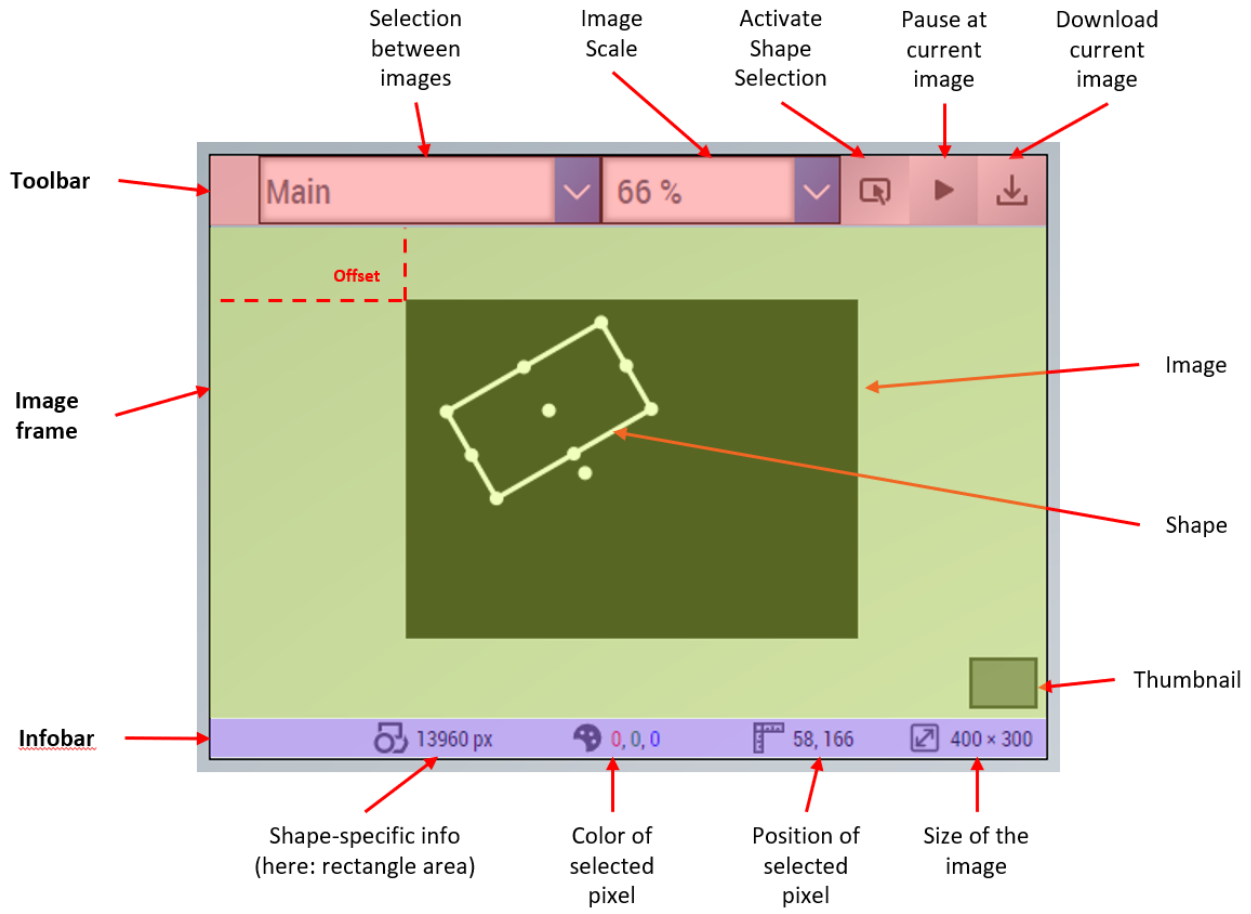
Themed Resources

Mit folgenden Attributen kann das Aussehen von Unter-Controls (den Comboboxen in der Toolbar) unter den Themes Einstellungen verändert werden. Eine detaillierte Beschreibung der einzelnen Attribute ist in der Standard-HMI-Dokumentation zu finden.

Attribut	Typ
ImageSelection_ContentPadding	Padding
ImageSelection_DataHeight	DataHeight
ImageSelection_DropDownBackgroundColor	Color
ImageSelection_DropDownFontFamily	FontFamily
ImageSelection_DropDownFontSize	MeasurementValue
ImageSelection_DropDownFontStyle	FontStyle
ImageSelection_DropDownFontWeight	FontWeight
ImageSelection_DropDownHorizontalAlignment	HorizontalAlignment
ImageSelection_DropDownTextColor	SolidColor
ImageSelection_DropDownVerticalAlignment	VerticalAlignment
ImageSelection_DropDownStyle	TcHmiCombobox.DropDownStyle
ImageSelection_FontFamily	FontFamily
ImageSelection_FontSize	Number (px)
ImageSelection_FontStyle	FontStyle
ImageSelection_FontWeight	FontWeight
ImageSelection_HorizontalAlignment	HorizontalAlignment
ImageSelection_MaxListHeight	Number (px)
ImageSelection_TextColor	SolidColor
ScaleSelection_ContentPadding	Padding
ScaleSelection_DataHeight	DataHeight
ScaleSelection_DropDownBackgroundColor	Color
ScaleSelection_DropDownFontFamily	FontFamily
ScaleSelection_DropDownFontSize	MeasurementValue
ScaleSelection_DropDownFontStyle	FontStyle
ScaleSelection_DropDownFontWeight	FontWeight
ScaleSelection_DropDownHorizontalAlignment	HorizontalAlignment
ScaleSelection_DropDownStyle	TcHmiCombobox.DropDownStyle
ScaleSelection_DropDownTextColor	SolidColor
ScaleSelection_DropDownVerticalAlignment	VerticalAlignment
ScaleSelection_FontFamily	FontFamily
ScaleSelection_FontSize	Number (px)
ScaleSelection_FontStyle	FontStyle
ScaleSelection_FontWeight	FontWeight
ScaleSelection_HorizontalAlignment	HorizontalAlignment
ScaleSelection_MaxListHeight	Number (px)
ScaleSelection_TextColor	SolidColor

7.2.1.1 Aufbau des Controls

Das **Image Watch** Control besteht aus Bereichen zur Anzeige von Werkzeugen und Informationen (Toolbar [▶ 2804] und Infobar [▶ 2805]), sowie einem Hauptbereich zur Anzeige des Bildes. Innerhalb dieses Bereichs kann das Bild programmatisch, per Maus oder per Touch verschoben und skaliert werden. Weitere Features wie das Thumbnail oder die Auswahl von Formen können applikationsspezifisch verwendet werden.



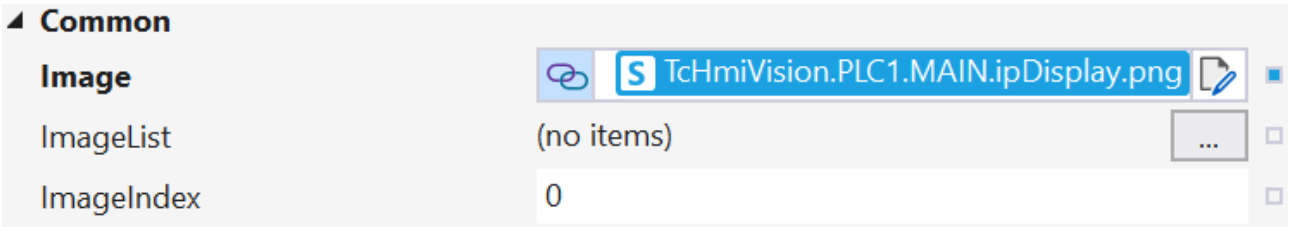
7.2.1.2 Bild-Ressourcen

Die auf dieser Seite erklärten Attribute und Funktionen sind in der Kategorie **Common** zu finden.

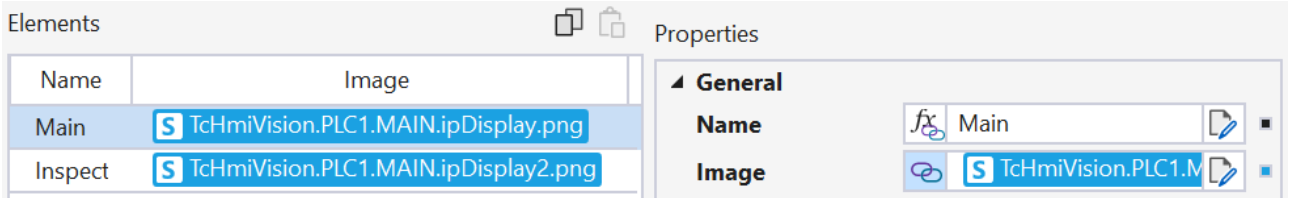
Bild-Ressourcen

Die Bild-Ressource definiert, welches Bild im Image Viewer angezeigt wird. Die beabsichtigte Benutzung ist, als Bild-Ressource ein `ITcVnDisplayableImage`-Symbol mittels der HMI.Vision Erweiterung [▶ 2784] zu verknüpfen. Im Allgemeinen muss die Bild-Ressource allerdings nur eine valide Bild-URL sein. Dies kann entweder ein direkter Pfad im Projekt, eine URL zu einem externen Server oder eine Data-URL sein. Dies funktioniert analog zum Src Attribut des Standard-Image-Controls.

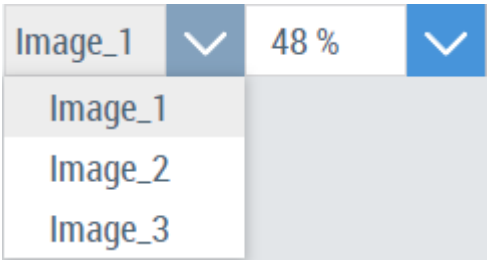
Im einfachsten Fall wird die Bild-Ressource über das `Image` Attribut festgelegt:



Darüber hinaus besteht die Möglichkeit, mehrere Bild-Ressourcen in dem `ImageList` Attribut zu hinterlegen:



Über das `ImageIndex` Attribut kann ausgewählt werden, welche der Ressourcen tatsächlich genutzt wird. Dies kann z. B. zum einfachen Umschalten zwischen Bild-Symbolen aus der PLC genutzt werden. Die Umschaltung kann zudem grafisch über die `ImageSelection` Combobox in der Toolbar [▶ 2804] erfolgen. Für diesen Zweck kann in der `ImageList` nicht nur die jeweilige Bild-Ressource, sondern auch ein zugehöriger Bild-Name vergeben werden.



Gleichzeitige Nutzung von Image und ImageList schließt sich gegenseitig aus

Intern führt die Verwendung der Eigenschaft `ImageList` dazu, dass das jeweilige Bildsymbol des ausgewählten `ImageList` Elements an das `Image`-Symbol gebunden wird. Daher schließen sich die Nutzung von `Image` und `ImageList` gegenseitig aus; sie sollten daher kein Bildsymbol an das `Image` Attribut binden, wenn Sie bereits `ImageList` verwenden.

Empfangsevent

Jedes Mal, wenn das Control ein neues Bild empfängt, wird das Event `onImageReceived` ausgelöst.

Alternativ-Text

Wenn keine Bild-Ressource festgelegt ist oder von der festgelegten Ressource kein valides Bild geladen werden konnte, wird ein Alternativ-Text angezeigt. Dieser kann durch das Attribut `Alt` verändert werden und ist standardmäßig „No image available.“.

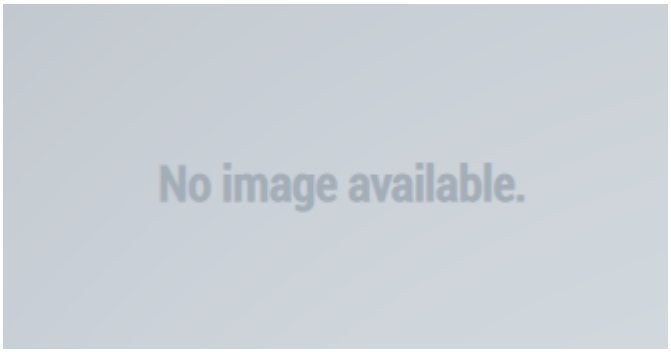


Bild-Download

Das aktuell angezeigte Bild kann mittels browserspezifischem Download lokal auf dem Client-Gerät gespeichert werden. Der Download kann entweder durch die Funktion `downloadImage` oder den Download-Button in der Toolbar [▶ 2804] ausgelöst werden.

Das Bild wird in Originalgröße gespeichert.

Der Dateiname kann bei programmatischen Aufrufen als Parameter angegeben werden. Eine Dateierweiterung muss nicht angegeben werden, da diese automatisch angehängt wird. Wenn der Parameter leer bleibt, wird der Dateiname ebenfalls automatisch erstellt. Bei einem Symbol wird zuerst der Symbolname verwendet, bei einer Data-URL entsprechend der Controlname:

<Symbolname/Controlname>_<Datum>_<Zeit>-<MS>.<Dateierweiterung>

Bei einem Bild-Pfad wird der Dateiname und die Dateierweiterung aus dem angegebenen Pfad verwendet, ohne einen Zeitstempel anzuhängen.

Die Auswahl des Speicher-Pfads ist innerhalb des Controls nicht möglich, sondern entspricht der Download-Pfad-Einstellung des genutzten Browsers. Der übliche Download-Pfad ist **C:\Benutzer\<Benutzer>\Downloads**.

Vorrausetzungen und Einschränkungen des Bild-Downloads

Da der Bild-Download über eine browserspezifische Funktionalität erfolgt, sind die Möglichkeiten, Einstellungen und die Funktionsweise abhängig vom verwendeten Webbrowser für den TwinCAT HMI Client.

Die Methode `downloadImage` erfordert die Berechtigung **operate** der TwinCAT HMI Benutzerverwaltung.

Diese Methode funktioniert nicht mit Bildern die als Stream verknüpft sind, sowie in der Live-Ansicht des HMI-Engineerings.

7.2.1.3 Bild-Ansicht

Die auf dieser Seite erklärten Attribute und Funktionen sind in der Kategorie **View** zu finden.

Intern ist die Größe und Position des angezeigten Bildes durch die Eigenschaften `Offset`, `Scale` und `ScaleReference` genau definiert. Der typische Anwendungsfall für die Änderung von Größe und Position ist jedoch die Verwendung von interaktiven Features wie Touch- und Mausgesten. Dabei werden die oben genannten Eigenschaften durch jene Gesten intern automatisch verändert.

Eigenschaft: `Offset`

Die Eigenschaft `Offset` gibt die Bildposition relativ zum Control selbst an. Die linke obere Ecke wird als Ursprung definiert, wobei die x-Achse nach rechts und die y-Achse nach unten verläuft. Der Versatz des Bildes wird ebenfalls von der oberen linken Ecke des Bildes aus gemessen. Die Offset-Koordinaten werden in Pixel angegeben.

Wie bereits erwähnt, wird die Eigenschaft `Offset` vom Control selbst geändert, z. B. wenn der Benutzer mit dem Bild durch Berührungs- oder Mausgesten interagiert.

Standardmäßig wird die Offset-Eigenschaft mit einem `TwoWay`-Binding verknüpft.

Eigenschaften: `Scale` **und** `ScaleReference`

Die Eigenschaft `Scale` gibt einen Skalierungsfaktor in Prozent an, um die Pixelgröße zu berechnen, in der das Bild auf dem Bildschirm angezeigt wird. Die Größe wird immer in Bezug auf die ursprüngliche Bildgröße oder bezogen auf die Größe des Controls berechnet, abhängig vom Wert des Attributs `ScaleReference`.

Die `ScaleReference` kann einer der folgenden Werte sein:

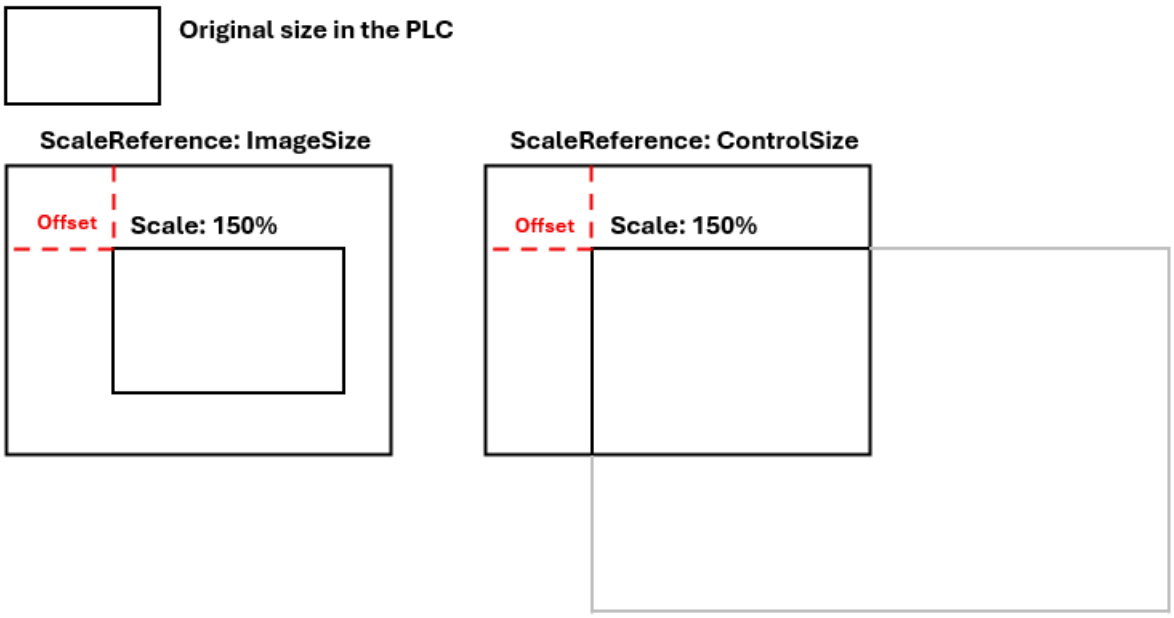
ImageSize	Die Größe des angezeigten Bildes entspricht der ursprünglichen Bildgröße multipliziert mit dem Wert der Eigenschaft `Scale`. Beispiel: Die Originalgröße des Bildes ist 300x200, die Eigenschaft `Scale` ist 150%. Die resultierende Größe des angezeigten Bildes ist 450x300.
ControlSize Width	Die Größe des angezeigten Bildes entspricht der Breite des Controls (oder genauer gesagt des Bildrahmens innerhalb des Controls) multipliziert mit dem Wert der Eigenschaft `Scale`.
ControlSize Height	Die Größe des angezeigten Bildes entspricht der Höhe des Controls (oder genauer gesagt des Bildrahmens innerhalb des Controls) multipliziert mit dem Wert der Eigenschaft `Scale`.
ControlSize	Die Größe des angezeigten Bildes entspricht der Größe des Controls (oder genauer gesagt des Bildrahmens innerhalb des Controls) multipliziert mit dem Wert der Eigenschaft `Scale`. Wenn das Seitenverhältnis des Controls nicht mit dem Seitenverhältnis des Bildes übereinstimmt, wird entweder die Höhe oder die Breite als Referenzgröße verwendet, je nachdem welche Abmessung stärker abweicht.

Der Vorteil der `ImageSize` Einstellung ist die direkte Beziehung zwischen der angezeigten Größe und der Originalgröße des Bildes. Zwei Bilder mit unterschiedlichen Größen in der SPS haben immer unterschiedliche Größen in der HMI. Der Vorteil der `ControlSize` Einstellung ist die automatische Anpassung des Bildes an das Control. Die Bilder haben immer die gleiche Größe, unabhängig davon, wie groß sie innerhalb der SPS tatsächlich sind.

Seitenverhältnis

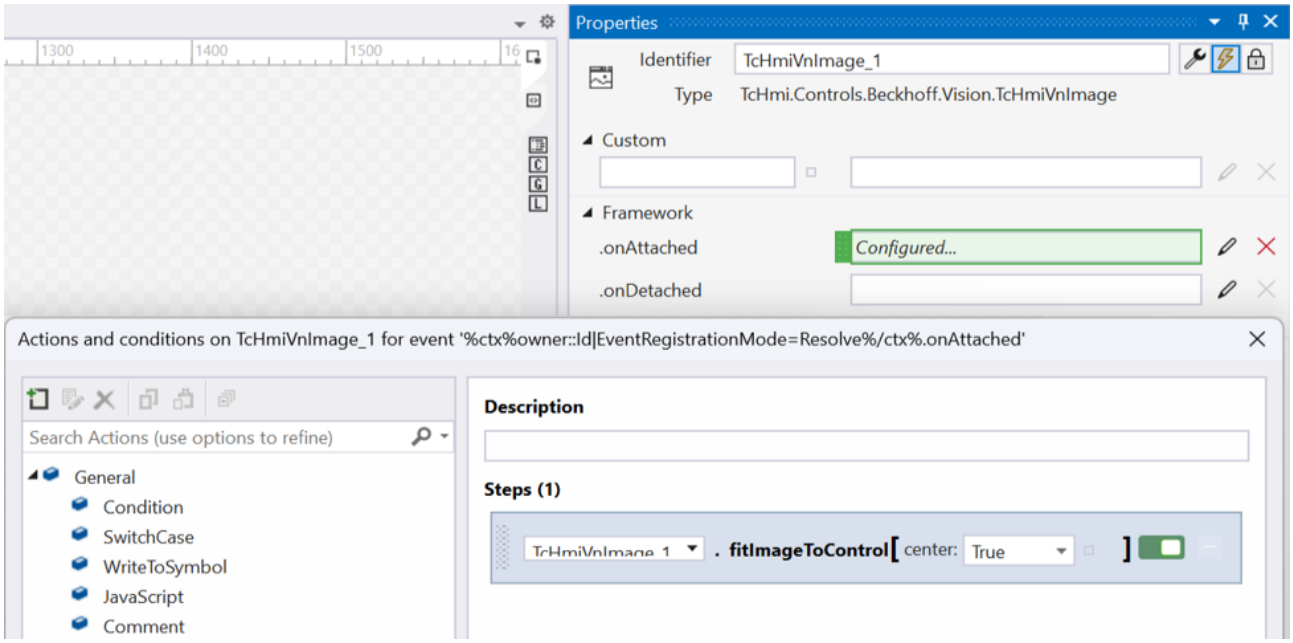
Höhe und Breite des Bildes werden immer so berechnet, dass das Seitenverhältnis des Originalbildes erhalten bleibt.

Beispiel: `Offset`, `Scale` **und** `ScaleReference`



Funktion: `FitImageToControl`

Um das aktuell angezeigte Bild innerhalb des Image Viewers zu maximieren und zu zentrieren, kann die Funktion FitImageToControl aufgerufen werden. Soll dies von Anfang an erfolgen, sollte die Funktion mit dem .onAttached Event verknüpft werden. Wenn diese Funktion einmal aktiviert wurde, werden alle folgenden Bilder ebenfalls zentriert und maximiert, bis die Ansicht anderweitig geändert und der Modus dadurch abgebrochen wird.



Eigenschaft: `ScalingCenter`

Das `ScalingCenter` bietet eine Möglichkeit, das Skalierungsverhalten einzustellen. Eine individuelle Änderung der `Scale`-Eigenschaften führt immer zu einer Veränderung der unteren rechten Ecke des Bildes, während die obere linke Ecke fixiert bleibt. Um dies zu ändern und andere Punkte zu fixieren, kann die Eigenschaft `ScalingCenter` verwendet werden. In diesem Fall wird die Eigenschaft `Offset` automatisch angepasst. Die folgenden Optionen sind verfügbar:

ScaleCenter-Wert	Beschreibung
`TopLeftCorner`	Obere linke Ecke des Bilds bleibt beim Zoomen zentriert.
`ImageCenter`	Mittelpunkt des Bilds bleibt beim Zoomen zentriert.
`ControlCenter`	Mittelpunkt des Controls bleibt beim Zoomen zentriert.

Gesten

Der Benutzer kann die Größe und Position des Bildes durch die folgenden Gesten/Interaktionen ändern:

- Halten Sie die linke Maustaste irgendwo im Bildrahmen und ziehen Sie die Maus, um das Bild zu verschieben und damit die Eigenschaft `Offset` zu ändern. Die Eigenschaft `Scale` bleibt davon unberührt.
- Berühren Sie den Bildrahmen mit einem Finger und bewegen Sie den Finger, um das Bild zu verschieben und damit die Eigenschaft `Offset` zu ändern. Die Eigenschaft `Scale` bleibt davon unberührt.
- Scrollen Sie mit dem Mausrad, um die Größe des Bildes und damit die Eigenschaft `Scale` zu ändern. Dabei wird der Mittelpunkt der Skalierung immer die Position des Mauszeigers sein. Infolgedessen kann sich auch die Eigenschaft `Offset` entsprechend ändern. Der Mauszeiger muss sich währenddessen immer über dem Control befinden.
- Berühren Sie den Bildrahmen mit zwei Fingern und bewegen Sie beide Finger in eine Richtung, zueinander hin oder voneinander weg, um die Position und Größe des Bildes zu ändern. Die beiden Eigenschaften `Offset` und `Scale` werden angepasst.

Um diese Benutzerinteraktionen zu verhindern, können die beiden Eigenschaften `Draggable` und `Scalable` auf `false` gesetzt werden, um eine benutzerinduzierte Änderung der Position bzw. Größe zu verhindern. Eine Änderung des Wertes (z. B. durch Schreiben auf das Symbol) ist immer möglich und wird durch die beiden Eigenschaften nicht beeinflusst.

Eigenschaften: `ScaleMin` und `ScaleMax`

Das Positionierungs- und Skalierungssystem des Controls wendet Begrenzungsregeln an, um zu verhindern, dass der Benutzer verwirrende oder desorientierende Ansichten des Bildes erzeugt.

Die Eigenschaften `ScaleMin` und `ScaleMax` stellen Grenzen für die Einstellung der `Scale`-Eigenschaft bei Benutzerinteraktionen dar.

Darüber hinaus beschränkt das Control automatisch das Bild auf seine Grenzen mit einem bestimmten äußeren Padding (derzeit 10px). Positionen außerhalb dieser Begrenzung sind nicht sinnvoll, da man das Bild nicht sieht.

Eigenschaft: `View` **(Schreibgeschützt)**

Die Eigenschaften `Offset` und `Scale` eignen sich gut für die interne Handhabung der Bildansicht, sind aber für den Anwender nicht unbedingt intuitiv (z. B., um einen Bildbereich zu definieren). Menschen denken oft eher in Form von Regionen von Interesse oder ähnlichen Darstellungen. Daher wird nach Änderung der Größe oder Position des Bildes der aktuell sichtbare Bereich des Bildes berechnet und über die schreibgeschützte Eigenschaft `View` bereitgestellt.

Die interne Berechnung setzt grundsätzlich den Koordinatenursprung auf die linke obere Bildecke und berechnet von dort aus die linke obere Position sowie die Breite und Höhe des Bildrahmens.

Die Eigenschaft `View` wird als Array in der Form `[X, Y, Breite, Höhe]` bereitgestellt.

Um die Position und Größe über eine View-Beschreibung festzulegen, siehe nächster Abschnitt über die Methode `SetView`.

Methode: `SetView`

Mit der Methode `SetView` kann der Benutzer einen bestimmten Bereich des Bildes anzeigen lassen. Beim Aufruf dieser Methode wird der durch einen Parameter angegebene Bildbereich maximiert und innerhalb des Bildrahmens zentriert.

Der Bereich muss in den ursprünglichen Bildkoordinaten als `[x, y, Breite, Höhe]` angegeben werden.

Abweichende Ansichten

Abhängig von den Seitenverhältnissen des angegebenen Bereichs und des Bildrahmens kann die geforderte Ansicht von der tatsächlichen Ansicht abweichen. Diese kann wiederum über die schreibgeschützte Eigenschaft `View` abgerufen werden.

Separate Aktion

Die View-Funktionalität ist nicht über eine Eigenschaft zugänglich, sondern über eine schreibgeschützte Eigenschaft und eine Setter-Methode. Der Grund dafür ist der oben erwähnte Unterschied zwischen einer bestellten und der tatsächlichen Ansicht und auch, dass eine **View**-Eigenschaft mit den `Offset`- und `Scale`-Eigenschaften darüber streiten würde, welche Ansicht erreicht werden soll. Dies wird dadurch verhindert, dass der View-Setter eine einmalige Aktion anstelle einer gebundenen Eigenschaft ist.

Bildfluss anhalten

Wenn die Eigenschaft `ImageFreeze` wahr ist, ändert sich das aktuell angezeigte Bild nicht, auch wenn neue Bilder empfangen werden. Der vorgesehene Anwendungsfall ist das Einfrieren des aktuellen Bildes zur weiteren Überprüfung, ohne dass zusätzliche Logik innerhalb der SPS hinzugefügt werden muss.

Wenn die Eigenschaft `ImageFreeze` von `true` auf `false` geändert wird, zeigt das Control sofort wieder das letzte Bild an und ignoriert die Zwischenbilder.

Die Eigenschaft `ImageFreeze` kann über den Freeze-Button in der Symbolleiste gesteuert werden (siehe Eigenschaft `FreezeButtonVisible` in der Kategorie Toolbar [▶ 2804]).

7.2.1.4 Toolbar

Die auf dieser Seite erklärten Attribute und Funktionen sind in der Kategorie **Toolbar** zu finden.

Über die Bedienelemente in der Toolbar können verschiedene Aktionen interaktiv ausgeführt werden, die sonst nur über die Attribute zugänglich sind. Jedes Element kann ein- oder ausgeblendet werden. Außerdem kann die gesamte Toolbar in ihrer Höhe angepasst oder vollständig ausgeblendet werden.

Die Positionierung der Elemente in der Toolbar ist fest und kann nicht individuell angepasst werden. Wenn solche Anpassungen dennoch erforderlich sind, können Sie die entsprechenden Elemente mittels der Eigenschaft `ToolbarItems` ausblenden und mit eigenen Controls selbst implementieren und auf der Image-Control Toolbar platzieren. Die verfügbaren Toolbar-Elemente von links nach rechts sind:

ImageSelection	Combobox zur Auswahl von Bildern aus der anzuzeigenden Bildliste (siehe Eigenschaft `ImageList` in der Kategorie **Common**). Wird nur angezeigt, wenn die Bildliste mindestens ein Element enthält. Beim Umschalten zwischen Bildern ist eine kurze Ladezeit zu erwarten.
ScaleSelection	Combobox zur Auswahl von bestimmten Skalierungswerten zur Anpassung des aktuellen Bildes. Die auswählbaren Skalierungswerte können über die Eigenschaft `ScaleOptions` angepasst werden. Wenn `ScaleSelectionEditable` deaktiviert wird, während eine benutzerdefinierte Skalierung eingestellt ist, zeigt die Combobox keinen Wert mehr an. Dies ist das Standardverhalten des Combobox-Controls.
ShapeButton	Schaltfläche zum Einstellen des Wertes der Eigenschaft `ScaleSelectionActive`. Wenn aktiv, wird ein vorhandenes Shape angezeigt und kann bearbeitet werden. Wird kein Shape angezeigt, kann ein neues gezeichnet werden.
FreezeButton	Schaltfläche zum Einstellen des Wertes der Eigenschaft `ImageFreeze`. Wenn aktiv, ist das aktuell angezeigte Bild festgesetzt und wird nicht mehr aktualisiert.
DownloadButton	Schaltfläche zum Herunterladen des aktuellen Bildes mit Standardnamen (siehe Funktion `downloadImage` unter Bild-Download [▶ 2799]). Um einen anderen Dateinamen zu vergeben, kann diese Schaltfläche selbst implementiert und dann die Funktion mit einem selbst definierten Namen aufgerufen werden.

Funktionen auch programmatisch zugänglich

Jede Aktion, die über die Oberflächenelemente der Toolbar zugänglich ist, ist auch programmatisch über die Eigenschaften des Controls zugänglich.

Größeneinstellungen

Die Toolbar bietet Anpassungsmöglichkeiten für die eigene Höhe über die Eigenschaft `ToolbarHeight` und für die Breite der beiden Comboboxen Eigenschaften `ImageSelectionWidth` und `ScaleSelectionWidth`. Die Breite der drei zusätzlichen Schaltflächen wird automatisch an die Höhe der Toolbar angepasst, so dass diese quadratisch sind.

Die Breite der Dropdown-Schaltfläche der Comboboxen kann nicht angepasst werden, da diese Funktionalität im Standard-Combobox-Control nicht vorhanden ist.

Intern wird die Eigenschaft `ToolbarHeight` auf maximal 150 Pixel begrenzt.

Hintergrundfarbe

Die Hintergrundfarbe der Toolbar ist standardmäßig ein halbtransparentes Weiß `"rgba(255, 255, 255, 127)"` und kann über die Eigenschaft `BarColor` in der Kategorie **Colors** angepasst werden.

Erweiterte Einstellungen

Weitere grafische Einstellungen (z. B. Schriftart der Comboboxen) können über Themed Resources verändert werden. Im Wesentlichen werden dabei Attribute der Comboboxen bereitgestellt. Die Themed Resources können im Theme-Editor unter `TcHmiVnImage > Themed Resources` angepasst werden. Diese Einstellungen gelten dann für alle Instanzen des Image Viewer Controls.

7.2.1.5 Infobar

Die auf dieser Seite erklärten Attribute und Funktionen sind in der Kategorie **Infobar** zu finden.

Über die Anzeigeelemente in der Infobar können verschiedene Informationen zu Shape-, Pixelwerten oder zur Bildgröße angezeigt werden. Jedes Element kann ein- oder ausgeblendet werden. Außerdem kann die gesamte Infobar in ihrer Höhe angepasst oder vollständig ausgeblendet werden.

Die Positionierung der Elemente in der Infobar ist fest und kann nicht individuell angepasst werden. Wenn solche Anpassungen dennoch erforderlich sind, können Sie die entsprechenden Elemente mittels der Eigenschaft **InfobarItems** ausblenden und mit eigenen Controls selbst implementieren und auf der Image-Control Infobar platzieren. Die verfügbaren Infobar-Elemente von links nach rechts sind:

ShapeValue	Berechneter Wert zu der aktuell angezeigten Form. Dieser Wert wird von der Funktion berechnet, die an die Eigenschaft `ShapeValueFormat` gebunden ist. Die Standardformatierungsfunktion [▶ 2834] `ShapeValueFormatter` gibt einen spezifischen Wert je nach ausgewähltem Formtyp zurück: • Koordinate von dem Punkt • Länge von der Linie • Fläche von dem Quadrat, Rechteck, Kreis oder Ellipse • Anzahl der Punkte des Polygons Dieses Bespiel [▶ 2806] beschreibt die Implementierung und Nutzung einer eigenen Formatter-Funktion.
PixelColor	Farbwert(e) des ausgewählten Pixels. Dieser Wert wird von der Funktion berechnet, die an die Eigenschaft `PixelColorFormat` gebunden ist. Die Standardformatierungsfunktion [▶ 2834] `PixelColorFormatter` gibt einen spezifischen Wert je nach ausgewähltem Farbraum zurück. Auswählbare Farbformate sind: `Gray, RGB, RGBa, HSV, HSV FULL, HLS, HLS FULL, Lab, YUV, YCrCb`. Des Weiteren kann eingestellt werden, ob und wie gerundet werden soll. Dabei kann zwischen 0-3 `Decimals` und `Unrounded` ausgewählt werden. Außerdem ist der PiyelType einstellbar. Mögliche Werte sind hier `8-Bit (USINT)` und `16-Bit (UINT)`. Dieses Bespiel [▶ 2807] beschreibt die Implementierung und Nutzung einer eigenen Formatter-Funktion.
PixelPosition	Position des ausgewählten Pixels.
ImageSize	Originalgröße des aktuell angezeigten Bildes.

Größeneinstellungen

Die Infobar bietet Anpassungsmöglichkeiten für die eigene Höhe über die Eigenschaft `InfobarHeight`, die intern auf maximal 150 Pixel begrenzt ist.

Hintergrundfarbe

Die Hintergrundfarbe der **Infobar** ist standardmäßig ein halbtransparentes Weiß "`rgba(255, 255, 255, 127)`" und kann über die Eigenschaft `BarColor` in der Kategorie **Colors** angepasst werden.

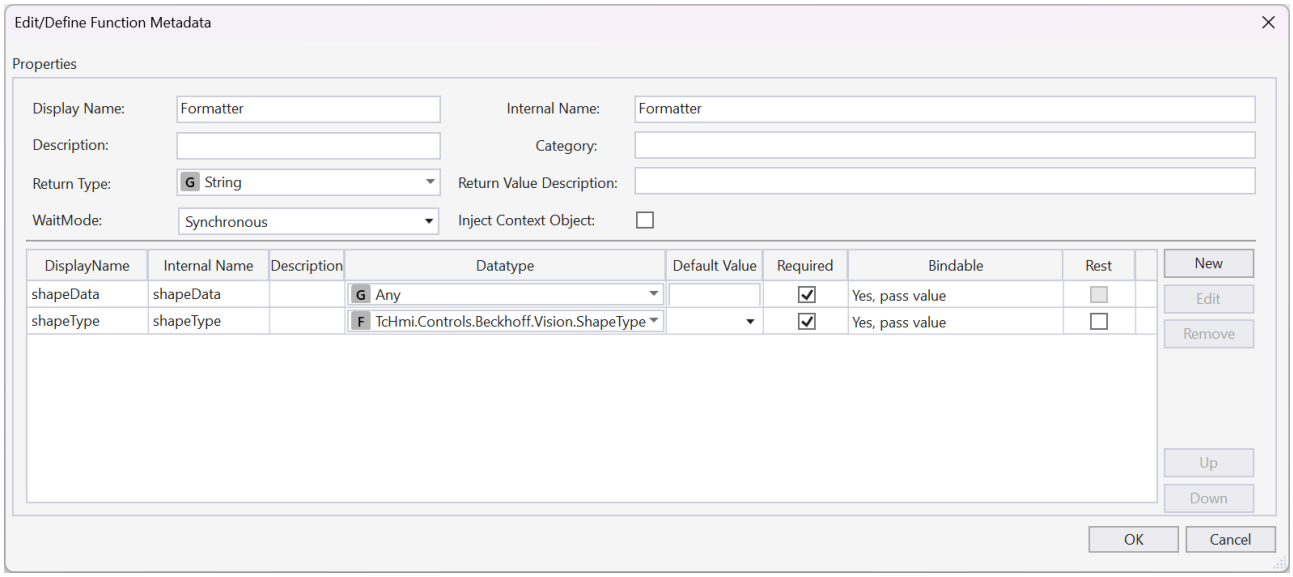
7.2.1.5.1 ShapeValueFormatting Funktion

Alternativ zur `ShapeValueFormatter` Funktion kann eine benutzerdefinierte Formatter-Funktion erstellt werden, um andere Shape Informationen in der Infobar anzuzeigen. Das folgende Beispiel zeigt, wie eine solche Funktion angelegt wird und welche Schnittstellen diese haben muss.

Die allgemeine Verwendung von Funktionen wird in der HMI-Dokumentation erklärt.

Nach dem Erstellen der TypeScript Funktion müssen zwei Parameter angelegt werden. Der erste bekommt die Shape-Beschreibung analog zur ShapeData-Eigenschaft übergeben, sowie sie auch vom Control abrufbar ist. Dieser Parameter muss als Typ `Any` angelegt werden. Der zweite Parameter bekommt den Shape-Typ analog zur ShapeType-Eigenschaft übergeben. Dieser Parameter muss als Typ `TcHmi.Controls.Beckhoff.Vision.ShapeType` angegeben werden. Es ist zu beachten, dass sich diese übergebenen Parameter zum Zeitpunkt der Funktionsausführung aus Timing-Gründen von den Werten der entsprechenden Control-Eigenschaften unterscheiden können.

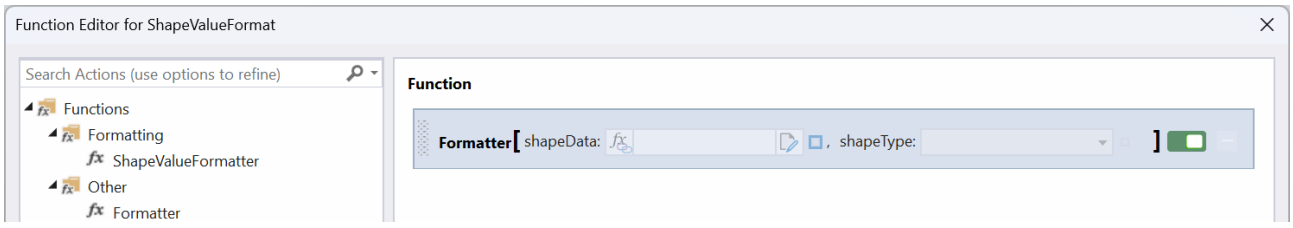
Die Funktion muss einen Wert vom Typ `String` zurückgeben, der dann in der Infobar angezeigt wird. Die Funktion wird immer aufgerufen, wenn sich die Shape-Beschreibung ändert und somit der String in der Infobar aktualisiert werden muss.



Im folgenden Codebeispiel wird die Breite und Höhe zurückgegeben, sofern die Objekte in den Formdaten enthalten sind:

```
namespace TcHmi {
    export namespace Functions {
        export namespace TcHmiProject {
            export function Formatter(shapeData: any, shapeType: any) {
                if (shapeData.stSize != null && shapeData.stSize.fWidth != null &&
                    shapeData.stSize.fHeight != null) {
                    return 'w: ' + shapeData.stSize.fWidth.toFixed(0) + ' h: ' +
                        shapeData.stSize.fHeight.toFixed(0);
                }
                else {
                    return '';
                }
            }
        }
    }
}
TcHmi.Functions.registerFunctionEx('Formatter', 'TcHmi.Functions.TcHmiProject',
TcHmi.Functions.TcHmiProject.Formatter);
```

Um eine selbst-erstellte Formatter-Funktion zu benutzen, klicken Sie bei der `ShapeValueFormat`-Eigenschaft auf den Auswahl-Button und wählen in dem Function Editor ihre soeben erstellte Funktion aus.



7.2.1.5.2 PixelColorFormatting Funktion

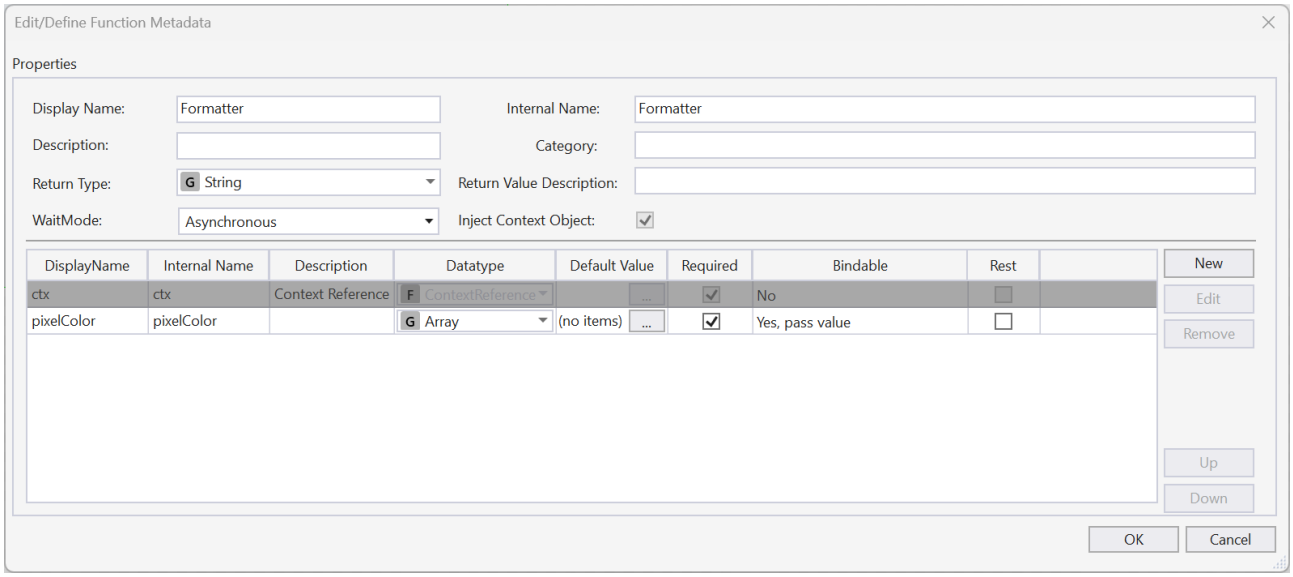
Alternativ zur `PixelColorFormatter` Funktion kann eine benutzerdefinierte Formatter-Funktion erstellt werden, um die Farbe des ausgewählten Pixels anders zu formatieren. Die Funktion erhält den Farbwert des Pixels und gibt einen String zurück, der in der Infobar angezeigt wird. Das folgende Beispiel zeigt, wie eine solche Funktion angelegt wird und welche Schnittstellen diese haben muss.

Die allgemeine Verwendung von Funktionen wird in der HMI-Dokumentation erklärt.

Nach dem Erstellen der TypeScript Funktion muss der WaitMode auf `Asynchronous` umgestellt werden. Das hat zur Folge, dass der Parameter „ctx“ angelegt wird, der für die Rückgabe notwendig ist. Danach muss ein weiterer Parameter vom Typ „Array“ angelegt werden, der analog zur PixelColor-Eigenschaft des

Controls den Farbwert als Array übergeben bekommt. Es ist zu beachten, dass sich der übergebene Parameter zum Zeitpunkt der Funktionsausführung aus Timing-Gründen vom Wert der entsprechenden Control-Eigenschaften unterscheiden kann.

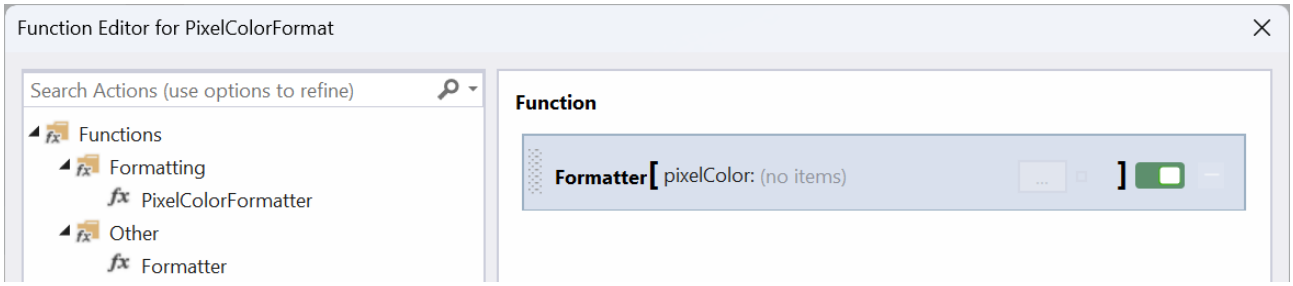
Die Funktion muss einen Wert vom Typ `String` zurückgeben, damit dieser in der Infobar angezeigt werden kann. Der Aufruf erfolgt immer, wenn sich der Wert der `PixelColor`-Eigenschaft ändert und somit der String in der Infobar aktualisiert werden muss.



Im folgenden Codeblock werden beispielhaft 3 Farbwerte in rot, grün und blau zurückgegeben, sofern mindestens 3 Elemente in pixelColor vorhanden sind. Dabei werden die Werte mit 256 multipliziert, um sie in 16-Bit umzurechnen:

```
namespace TcHmi {
    export namespace Functions {
        export namespace TcHmiProject {
           export function Formatter(ctx: any, pixelColor: number[]) {
                if (pixelColor.length >= 3) {
                    ctx.success(`<span style="color:red;">${pixelColor[0] * 256}</span>, ` +
                                `<span style="color:green;">${pixelColor[1] * 256}</span>, ` +
                                `<span style="color:blue;">${pixelColor[2] * 256}</span>`);
                }
                else {
                    ctx.success('');
                }
            }
        }
    }
}
TcHmi.Functions.registerFunctionEx('Formatter', 'TcHmi.Functions.TcHmiProject',
TcHmi.Functions.TcHmiProject.Formatter);
```

Um eine selbst erstellte Formatter-Funktion zu benutzen, klicken Sie bei der `PixelColorFormat`-Eigenschaft auf den Auswahl-Button und wählen in dem Function Editor ihre soeben erstellte Funktion aus.



7.2.1.6 Formen

Die auf dieser Seite erklärten Attribute und Funktionen sind in der Kategorie **Shapes** zu finden.

Ein häufiger Anwendungsfall ist die Auswahl bestimmter Bildbereiche durch den Benutzer. Dies kann eine Region of Interest oder die Position eines Objekts sein. Ein anderer verwandter Anwendungsfall ist die benutzerdefinierte Messung von Entfernungen oder Bereichen innerhalb eines Bildes.

Um all diesen Anwendungsfällen Rechnung zu tragen, bieten die Eigenschaften in der Kategorie **Shapes** eine flexible Funktionalität zum Zeichnen und Bearbeiten verschiedener Formen auf dem Bild und zum Abrufen ihrer jeweiligen geometrischen Beschreibungen.

Eigenschaft: `ShapeSelectionActive`

Die Auswahl einer Form kann über die Eigenschaft `ShapeSelectionActive` bzw. über den **Shape**-Button in der Toolbar [▶ 2804] aktiviert werden. Solange diese Eigenschaft aktiv ist, kann die Form durch Klicken auf die Bildoberfläche gezeichnet und durch Ziehen und Ablegen der Eckpunkte oder der Form selbst bearbeitet werden. Wenn diese Eigenschaft auf `false` gesetzt wird und die Eigenschaft `ShapeAutoClear` aktiv ist, wird die aktuelle Form zurückgesetzt und geht verloren; eventuell benötigte Daten müssen also vorher extrahiert werden.

Die geometrische Beschreibung der gezeichneten Form kann über die Eigenschaft `ShapeData` abgerufen werden. Je nach der Eigenschaft `ShapeType` (siehe unten) haben die Formdaten ein anderes Format. Sie können genau das gleiche Format erwarten, das in den interaktiven Controls selbst verwendet wird (z. B. `TcHmiVnPolygon` und `TcHmiVnRectangle`), da das Symbol direkt an das intern verwendete Shape Control gebunden ist. Mit der Eigenschaft `ShapeLimitToImage` kann die Form auf den Bildbereich beschränkt werden. Wenn dies nicht der Fall ist (default), ist zu beachten, dass die Formen auch negative Werte annehmen können. Mit der Funktion `shapeIsInImageArea` kann abgefragt werden, ob sich die Form innerhalb des Bildes befindet.

Eigenschaften: `ShapeType und ShapeData`

Es stehen verschiedene Arten von Shapes zur Verfügung, die auf zwei verschiedenen Datentypen basieren. Sie können über die Eigenschaft `ShapeType` eine der folgenden Formen auswählen:

ShapeType	ShapeData	PLC Declaration
Point	[[x, y]]	`ARRAY[0..0] OF TcVnPoint2_REAL or ARRAY[0..0] OF TcVnPoint2_LREAL`
Line	[[x1, y1], [x2, y2]]	`ARRAY[0..1] OF TcVnPoint2_REAL or ARRAY[0..1] OF TcVnPoint2_LREAL`

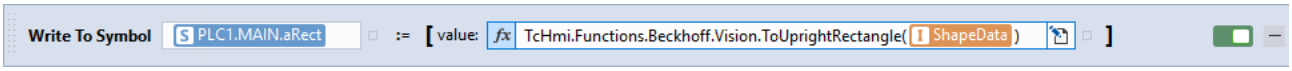
Square,
Rectangle,
Circle,
Ellipse
	{
	fAngle: angle,
	aCenter: [cx, cy],
	stSize: {
	fWidth: width,
	fHeight: height,
	},
	}
		`TcVnRotatedRectangle`

Polygon	[[x1, y1], [x2, y2], …, …, [xn, yn]]	`ARRAY[0..n] OF TcVnPoint2_REAL or ARRAY[0..n] OF TcVnPoint2_LREAL`

Die Werte in `ShapeData` beziehen sich auf die Bildkoordinaten des Original-Bilds.

Da die Shape-Typen Quadrat und Kreis per Definition immer die gleiche Höhe und Breite haben, werden sie beim Anklicken der zweiten Position nicht genau an diesen Endpunkt gezeichnet, wenn das Seitenverhältnis nicht gleich ist. In diesem Fall kann die gewünschte Position durch Verschieben der entsprechenden Ecke erreicht werden.

Um die Daten der Shape-Typen Quadrat und Rechteck vom Typ TcVnRotatedRectangle [▶ 2836] direkt an eine SPS-Variable vom Typ TcVnRectangle_UDINT [▶ 258] oder ein TcVnRectangle [▶ 2836] übergeben und als aufrechtes Begrenzungsrechteck ohne Drehwinkel verwenden zu können, steht die Konvertierungsfunktion ToUprightRectangle [▶ 2835] zur Verfügung. Für den umgekehrten Fall gibt es die ToRotatedRectangle [▶ 2834] Funktion.



Programmatisch kann die `ShapeData`-Eigenschaft per Java Skript wie folgt verändert werden:

```
let control = TcHmi.Controls.get("TcHmiVnImage")
// Polygon with 4 Points
control.setShapeData([[10, 10], [10, 100], [150, 150], [100, 10]])

// Rectangle
control.setShapeData({"aCenter":[200,200],"fAngle":15,"stSize":{"fWidth":150,"fHeight":150}})
```

Im ersten Beispiel wird ein Polygon mit einem Array von 4 Punkten geschrieben. Zum Schreiben eines Punkts benötigt man nur das erste Element und für eine Linie die ersten beiden. Im zweiten Beispiel wird ein Rechteck geschrieben. Da die Shape Typen Quadrat, Kreis und Ellipse auf dem gleichen Datentyp basieren, kann die gleiche Schreibweise auch für diese Typen verwendet werden.

Die Daten können auch per WriteToSymbol-Aktion an `ShapeData` geschrieben werden. Dazu kopieren Sie den Teil innerhalb der runden Klammern der `setShapeData()` Funktion und fügen ihn als Objektwert ein.

Eigenschaft: `ShapeAutoClear`

Wenn die Eigenschaft `ShapeAutoClear` aktiv ist und der `ShapeType` geändert wird, während `ShapeSelectionActive` `true` ist, wird die aktuelle Form zurückgesetzt und eine neue Form des ausgewählten Formtyps kann gezeichnet werden. Wenn `ShapeAutoClear` nicht aktiv ist und zwischen Formtypen des gleichen Basis-Controls umgeschaltet wird, bleiben die Daten beim Umschalten des `ShapeType` erhalten und es wird direkt die neue Form angezeigt. Die Typen Punkt, Linie und Polygon gehören dabei zu einer Gruppe die das gleiche Basis-Control verwenden, die Typen Quadrat, Rechteck, Kreis und Ellipse zu der anderen.

Erscheinungsbild

Das Aussehen der Form und der Bedienpunkte kann durch Einstellen der Farbe und der Größe/Dicke der Kontur geändert werden:

- Mit `ShapeStrokeThickness` kann die Dicke der Linien in Pixeln eingestellt werden.
- Mit `ShapeHandleSize` kann die Größe der angezeigten Bedienpunkte der Form in Pixeln eingestellt werden
- Mit `ShapeClickableSize` kann die Größe der klickbaren Fläche eines Bedienpunkts in Pixeln eingestellt werden. Das kann die Bedienung auf Touch-Panels vereinfachen, ohne die Handles unnötig groß anzeigen zu müssen.
- Über `ShapeStrokeColor` und `ShapeHandleColor` kann die Farbe der Linien und die der Bedienpunkte unabhängig voneinander eingestellt werden.

Eigenschaft: `ShapeIsRotatable`

Wenn `ShapeIsRotatable` aktiv ist, wird bei geeigneten Formen (Linie, Rechteck, Quadrat, Ellipse, Polygon) ein zusätzliches Handle angezeigt, mit dem die Form rotiert werden kann. Über die `ShapeAngleInterval` Eigenschaft kann die Schrittgröße, in der die Rotation der Form geändert werden kann, in Grad eingestellt werden. Bei dem Defaultwert 0° kann die Form beliebig gedreht werden. Wenn die Form z. B. nur in 4 Positionen platziert werden darf, würde man 90° als Wert eintragen.

Eigenschaft: `ShapeLimitToImage und ShapeIsInImageArea`

Häufig ist es erforderlich, dass eine Form vollständig innerhalb des Bildes liegt, z. B.,um eine Region of Interest setzen zu können. Wenn die Eigenschaft `ShapeLimitToImage` aktiv ist, kann eine Form nur innerhalb des Bildes gezeichnet und bewegt werden. Diese Einschränkung wirkt sich nicht auf das programmatische Setzen einer Form aus, sondern lediglich auf grafische Interaktionen. Weiterhin ist zu beachten, dass wenn `ShapeIsRotatable` und `ShapeAngleInterval` gesetzt sind, die Form weiterhin uneingeschränkt gedreht werden kann.

Wenn `ShapeLimitToImage` nicht aktiv ist, kann die Positionierung der Form über die Eigenschaft `ShapeIsInImageArea` abgefragt werden. Diese gibt nur dann `true` zurück, wenn aktuell valide `ShapeData` vorliegen und wenn alle Eckpunkte der Form innerhalb des Bildbereichs liegen. Bei den Formen Circle und Ellipse werden ebenfalls die Eckpunkte zur Überprüfung verwendet. Um zwischen einem nicht vorhandenen und einem nicht im Bild liegenden Shape unterscheiden zu können, sollte zusätzlich abgefragt werden, ob `ShapeSelectionActive` gesetzt und `ShapeData` valide ist.

Eigenschaft: `ShapeDataUpdate`

Wann eine Aktualisierung der Eigenschaft `ShapeData` erfolgen soll, kann über die Eigenschaft `ShapeDataUpdate` festgelegt werden. Wenn `onChanged` ausgewählt ist, wird `ShapeData` während einer grafischen Interaktion fortwährend aktualisiert. Wenn hingegen `onConfirmed` ausgewählt ist, wird `ShapeData` erst am Ende einer Interaktion aktualisiert.

Eigenschaften: `ShapePolygonNumPoints und ShapePolygonClosed`

Wenn die Eigenschaft `ShapeType` auf `Polygon` eingestellt ist, kann die Anzahl an Punkten ab wann ein Polygon als abgeschlossen betrachtet wird, über die Eigenschaft `ShapePolygonNumPoints` festgelegt werden. Wenn beispielsweise drei Punkte angegeben werden, wird das Polygon beim Zeichnen des dritten Punkts automatisch vervollständigt. Wenn der Defaultwert 0 verwendet wird, ist die Anzahl an Punkten beliebig. Das Polygon wird dann erst beim Klick auf den ersten Punkt vervollständigt. Mit der Eigenschaft `ShapePolygonClosed` kann zudem festgelegt werden, ob der erste und der letzte Punkt des Polygons verbunden werden sollen, also ob das Polygon geschlossen angezeigt wird oder nicht.

Methode: `clearShape()`

Die gezeichnete Form kann programmatisch durch Aufruf der Methode `clearShape` gelöscht werden.

Anzeigeelement

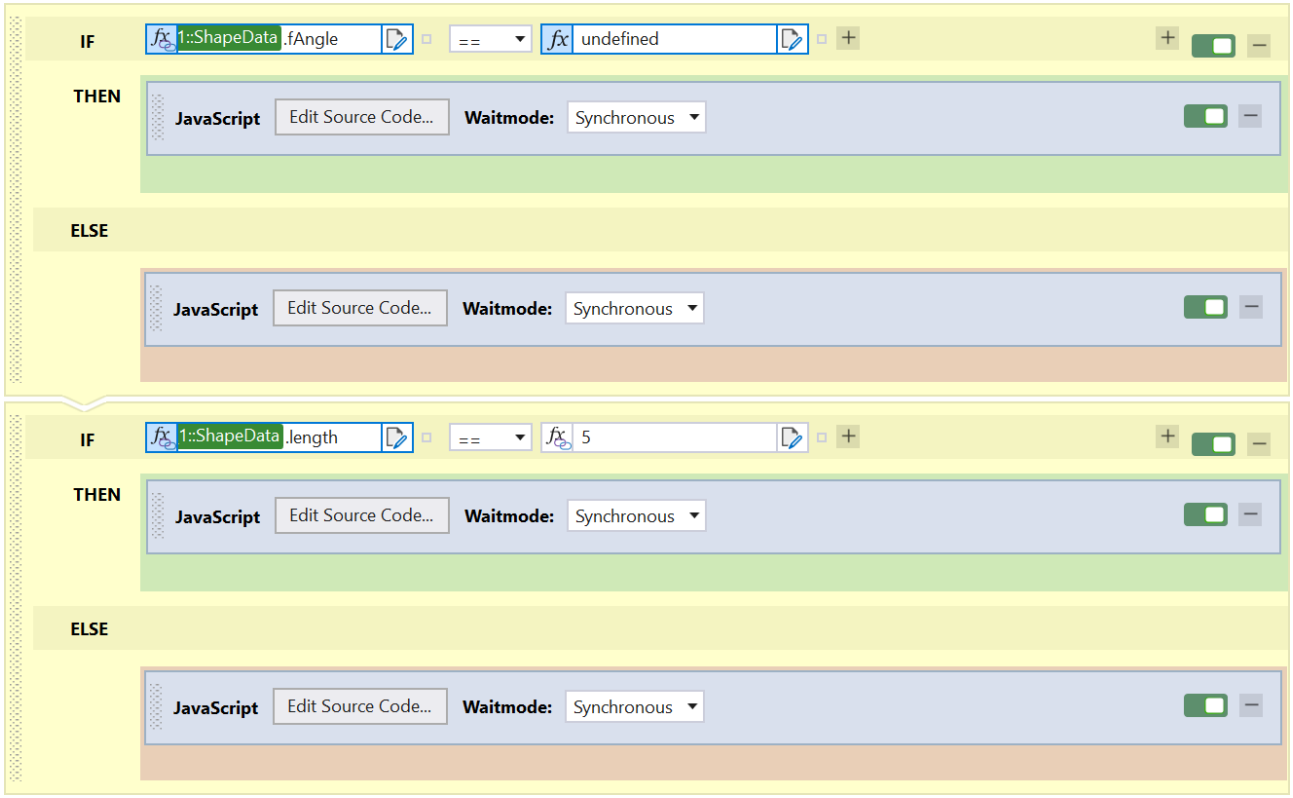
Um den Anwendungsfall der Messung von Entfernungen oder Flächen innerhalb des Bildes zu bedienen, bietet die Infobar [▶ 2805] ein Element zur Anzeige von Text, der die aktuell gezeichnete Form repräsentiert. Standardmäßig wird für den Formtyp "Punkt" die Koordinate, für "Linie" die Linienlänge, für "Polygon" die Anzahl der Punkte und für die Formtypen "Quadrat ", "Rechteck", "Kreis" und "Ellipse" die Fläche angezeigt. Sollen weitere oder andere Informationen angezeigt werden, kann dies mithilfe einer eigenen ShapeValueFormatting Funktion [▶ 2806] realisiert werden.

Abfrage der Formtypen und Datenwerte

Abhängig vom ausgewählten Formtyp können die Daten in `ShapeData` verschiedene Formate haben. Um festzustellen welches Format vorliegt, kann entweder der aktuelle `ShapeType` abgefragt oder anhand der Datenwerte ermittelt werden, z. B. ob die Eigenschaft fAngle definiert ist. Wenn das der Fall ist, liegt ein TcVnRotatedRectangle Typ vor, ansonsten ein Array von Punkten. Um abzufragen, wie viele Punkte im Array enthalten sind, kann die Länge abgefragt werden. Über die Anzahl kann dann festgestellt werden, ob ein Punkt, eine Linie oder ein Polygon mit welcher Anzahl von Punkten vorliegt.

Weiterhin kann durch Abfrage der einzelnen Werte geprüft werden, ob valide Daten vorliegen. Beim Löschen oder Zurücksetzen der Formdaten werden die einzelnen Werte mit 0 beschrieben, was einen ungültigen Wert darstellt.

Über diese Abfragen kann sichergestellt werden, dass die verschiedenen Formtypen jeweils an eine dafür vorgesehene SPS-Variable vom gleichen Typ geschrieben werden und dass nur gültige Daten in das Steuerungsprogramm übernommen werden können.



Events: `onShapeChanged` und `onShapeConfirmed`

Um über eine Änderung der Form informiert zu werden, stehen die beiden Events `onShapeChanged` und `onShapeConfirmed` zur Verfügung.

Das Event `onShapeChanged` wird ausgelöst, sobald sich `ShapeData` ändert. Zu einer Änderung kann es beim programmatischen Setzen von `ShapeData`, beim Aufruf der Methode `clearShape()` und während grafischer Interaktionen kommen. Dieses Event ist dazu da, die Änderung von `ShapeData` live zu verfolgen, vorausgesetzt, dass `ShapeDataUpdate` auf den Wert `onChanged` gesetzt ist.

Das Event `onShapeConfirmed` wird ausgelöst, wenn das Shape programmatisch gesetzt oder nachdem eine grafische Interaktion mit dem Shape abgeschlossen wurde. Dieses Event wird unter der Voraussetzung ausgelöst, dass `ShapeData` valide ist.

Erweiterte Nutzung von Shapes

Die Shape-Funktionalität wird über separate Controls implementiert, die in das Image Watch Control eingebunden sind. Diese Shape-Controls besitzen einige Properties, die nicht direkt über das ImageWatch Control verfügbar sind. Dazu zählt z. B. das exakte Seitenverhältnis eines Rechtecks. Auf diese Properties kann entweder programmatisch oder durch separate Verwendung der Shape-Controls zugegriffen werden.

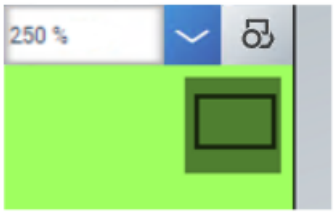
7.2.1.7 Thumbnail

Die auf dieser Seite erklärten Attribute und Funktionen sind in der Kategorie **Thumbnail** zu finden.

Der Thumbnail zeigt an, welcher Ausschnitt des Bildes im Control sichtbar ist. Er steht in engem Zusammenhang mit der **Ansicht** des Bildes (siehe Read-Only-Eigenschaft `View` in der Kategorie **View**), unterscheidet sich aber in einigen Punkten.

Ob der Thumbnail sichtbar ist, wird durch die Eigenschaft `ThumbnailVisbile` bestimmt. Seine allgemeine Position und Größe wird durch die Eigenschaften `ThumbnailPosition` und `ThumbnailSize` bestimmt und ist immer quadratisch. Wenn das Seitenverhältnis des Bildes nicht 1:1 ist, entspricht die Größe des Thumbnails der vollen Bildbreite, während die Höhe an das Seitenverhältnis angepasst wird.

Wenn der gesamte Bildrahmen mit dem Bild bedeckt ist, ist die Darstellung des Thumbnails praktisch identisch mit der schreibgeschützten Eigenschaft `View`. Andernfalls unterscheidet sie sich dahingehend, dass Bereiche außerhalb des Bildes in der `View` berücksichtigt werden, aber nicht im Thumbnail.



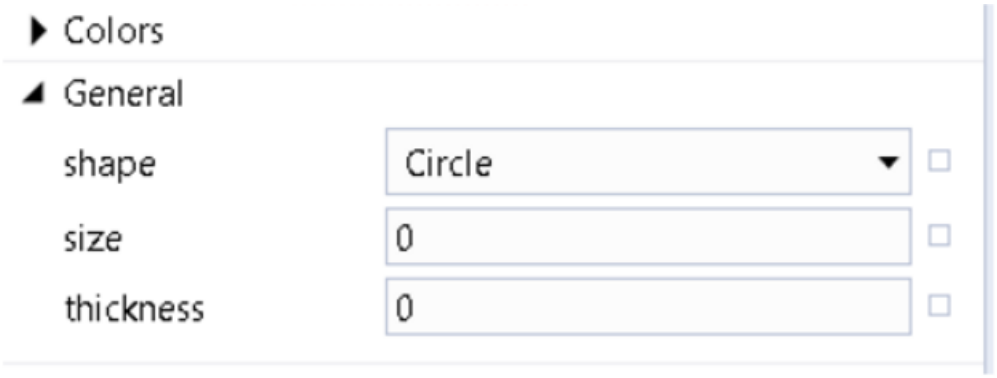
7.2.1.8 Einblendungen

Die auf dieser Seite erklärten Attribute und Funktionen sind in der Kategorie **Overlay** zu finden.

Einblendungen können genutzt werden, um z. B. die Einrichtung der Kamera oder das Positionieren von Objekten innerhalb des Bildes zu erleichtern.

Vordefinierte geometrische Formen

Das Control bietet vordefinierte geometrische Objekte (wie Kreise, Rechtecke, Kreuze, Rahmen), die in der Mitte des Bildes angezeigt werden können. Diese Formen können z. B. als Fadenkreuz oder für Rahmen und Markierungen an bestimmten Positionen im Bild dienen. Es können auch mehrere Objekte gleichzeitig angezeigt werden (z. B. ein Kreis und ein Fadenkreuz-Cursor). Dies geschieht über die Array-Eigenschaft `OverlayElements`. Die Konfiguration eines Objekts sieht wie folgt aus:



Die Objekte werden immer im Bild zentriert. Sie haben einen festen Maßstab, der über **Größe** definiert wird und skalieren nicht mit dem Bild.

Overlay-Bild

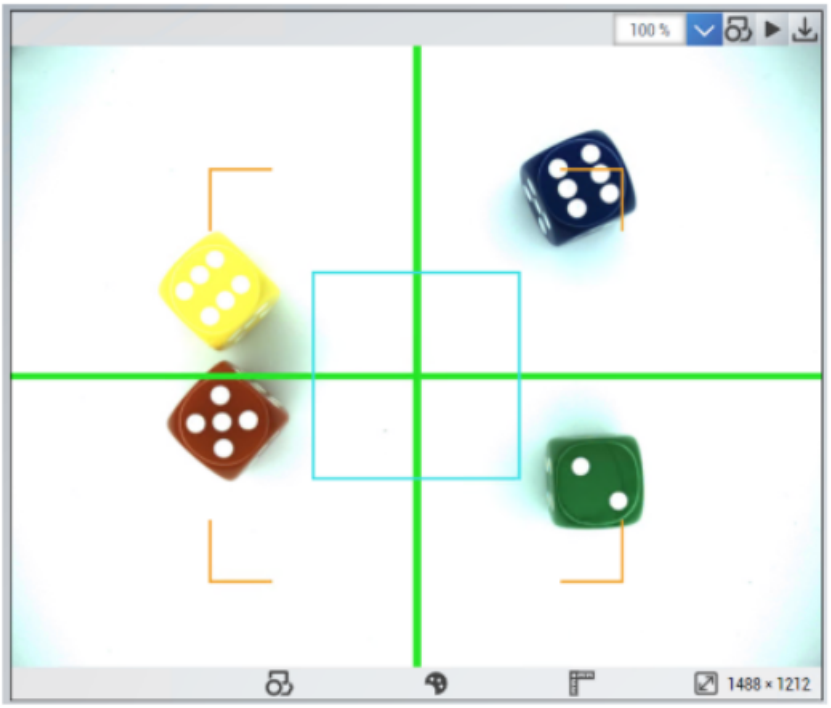
Wenn diese Funktionalität für eine bestimmte Anwendung nicht flexibel genug ist, können Sie über die Eigenschaft `OverlaySource` ein Overlay-Bild festlegen. Dieses wird an genau der gleichen Stelle und in der gleichen Größe wie das Hauptbild angezeigt, d.h. das Overlay-Bild wird gestreckt/gequetscht, um es an das Hauptbild anzupassen. Wir empfehlen daher, dass das Overlay-Bild mindestens das gleiche Seitenverhältnis und vorzugsweise die gleiche Größe wie das Hauptbild hat und bis auf relevante Teile transparent ist.

Als Beispiel könnten Sie einen benutzerdefinierten Fadenkreuz-Cursor mit zusätzlichen Accessoires auf ein transparentes Bild zeichnen und dieses als Overlay verwenden. Um ein transparentes Bild zu erzeugen, muss ein 4 kanaliges Bild mit dem Farbwert 0 beim vierten Kanal der Hintergrundfarbe erstellt werden. Die einzuzeichnenden Cursor-Linien bekommen dann den Farbwert 255 beim vierten Kanal, so dass diese Elemente das Hintergrundbild überlagern.

Ansicht

Ein Overlay-Bild mit einem Rechteck und einem Text könnte zum Beispiel das folgende Ergebnis liefern:

Im folgenden Beispiel wurde ein Overlay-Bild aus der SPS (grüne Linien) mit zwei Objekten kombiniert, die im Control definiert wurden:



Dabei ist zu beachten, dass bei der Übertragung von transparenten Bildern mit der „TwinCAT.HMI.Vision“ Server Extension das Bildformat PNG oder BMP verwendet werden muss, da JPEG und Stream keine transparenten Bildinhalte mit Alphakanal unterstützt.

7.2.1.9 Infos

Die auf dieser Seite erklärten Attribute und Funktionen sind in der Kategorie **Info** zu finden.

Mit folgenden Attributen können Sie Informationen über das angezeigte Bild oder spezifische Pixel in diesem Bild abfragen.

ImageSize	Originalgröße des angezeigten Bildes. Diese entspricht der Bildgröße in der SPS.
ImageDisplaySize	Angezeigte Größe des Bildes in Pixeln.
PixelColor	Farbe eines bestimmten ausgewählten Pixels (siehe unten für Details). Die Farbe wird als Array mit vier Elementen bereitgestellt, die die Farbkanäle [R, G, B, A] darstellen.
PixelPosition	Position eines bestimmten ausgewählten Pixels (siehe unten für Details). Die Position wird als Array mit zwei Elementen [X, Y] in Originalbildkoordinaten, nicht in Anzeigekoordinaten, angegeben.

Die **PixelColor** und **PixelPosition** werden für ein bestimmtes Pixel berechnet. Die Auswahl dieses Pixels erfolgt durch einen Klick, eine Berührung oder durch die Position des Cursors auf dem angezeigten Bild.

Die Eigenschaft **PixelInfoUpdate** bestimmt, ob die Berechnung überhaupt durchgeführt wird oder durch welches Ereignis sie ausgelöst wird.

Never	**PixelColor** und **PixelPosition** werden überhaupt nicht aktualisiert.
AtClick	**PixelColor** und **PixelPosition** werden aktualisiert, wenn der Benutzer auf ein bestimmtes Pixel klickt oder es berührt. Die Werte werden zurückgesetzt, wenn der Benutzer auf einen Punkt des Bildrahmens außerhalb des aktuellen Bildes klickt oder diesen berührt.
AtCursor	**PixelColor** und **PixelPosition** werden kontinuierlich aktualisiert, während der Benutzer den Mauszeiger über das Bild bewegt. Die Werte werden zurückgesetzt, wenn sich der Mauszeiger nicht auf dem Bild befindet.

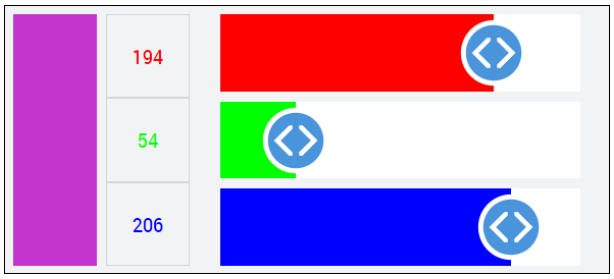
Jedes Mal, wenn die Pixelfarbe und -position aktualisiert werden, wird das Ereignis **onPixelInfo** ausgelöst.

Anzeige

Die aktuellen Informationen von PixelColor, PixelPosition und ImageSize können in der Infobar angezeigt werden (siehe Eigenschaft `InfobarItems` in der Kategorie **Infobar**).

7.2.2 Color

Das Color Control zeigt Farben an und macht diese bei Bedarf einstellbar. Es enthält verschiedene Anzeigeelemente, um die Farbwerte zu repräsentieren, die abhängig von der Konfiguration auch zum Einstellen der Farbwerte verwendet werden können. Eine Übersicht über die Komponenten und den Aufbau finden Sie im Unterkapitel Anzeigeelemente [▶ 2817]. Die Beschreibung der einzelnen Konfigurationsmöglichkeiten finden Sie unter den folgenden Kategorien auf dieser Seite. Zu den jeweiligen Kategorien gibt es Unterkapitel mit weiteren Informationen, Details und Anwendungsbeispielen.



Für alle folgenden Attribute besitzt das Control eine Getter- und eine Setter-Methode. Zum Beispiel existieren für das Attribut `Orientation` die beiden Methoden `getOrientation` und `setOrientation`. Die Benutzung der Attribute wird in den verlinkten Unterkapiteln näher erklärt.

Alle weiteren Attribute, Events und Permissions die die Basisklasse **Controls** bereitstellt, finden Sie in der TwinCAT HMI Dokumentation.

Kategorie: Common

Die folgenden Attribute stellen eine Datenschnittstelle für den Farbwert [▶ 2819] des Controls dar. Es wird zum einen benutzt, um die angezeigte Farbe des Controls einzustellen, zum anderen um einen vom Nutzer eingegebenen Farbwert abzurufen.

Attribut	Typ	Beschreibung
`ColorSymbol`	ColorSymbol [▶ 2835]	HMI-Symbol, welches den aktuellen Farbwert enthält.
`ColorChannels`	ColorChannels [▶ 2835]	Definiert das Format der Farbe. Dies beinhaltet: • Anzahl der Kanäle • Min-/Max-Werte der Kanäle • Integer oder Float Anzeige-Farben der Kanäle
`BoxColorConversion`	`Function`	Funktion, die den jeweiligen Farbwert in einen vom Browser interpretierbaren String konvertiert und umgekehrt. Dies ist nötig, um dem Control die Bedeutung der einzelnen Farbkanäle mitzuteilen, sofern diese vom Standardwert RGB-Format abweicht.
`ColorValue` (readonly)	ColorValue [▶ 2836]	Farbwert. Dieser kann abgerufen werden, ohne dass ein Symbol als `ColorSymbol` verknüpft ist.

Das Attribut `ColorValue` passt sich immer an den Datentype der gemappten SPS-Variable an. Standard ist ein 4 kanaliges Array (`TcVnVector4_LREAL`) was auch ohne ein Mapping verwendet wird. Weitere Möglichkeiten siehe Verknüpfung mit PLC [▶ 2820].

Funktion	Beschreibung
`setColorValue(colorValue:` ColorValue [▶ 2836]`): void`	Setzt den Farbwert. Kann aufgerufen werden, ohne dass ein Symbol als `ColorSymbol` verknüpft ist.

Kategorie: Usage

Die folgenden Attribute legen das Verhalten des Controls fest.

Attribut	Typ	Beschreibung
`BoxUsage`	UsageMode [▶ 2836]	Bestimmt, ob die Farbbox angezeigt wird und ob sie bearbeitbar ist.
`ValuesUsage`	UsageMode [▶ 2836]	Bestimmt, ob die einzelnen Farbwerte angezeigt werden und ob sie bearbeitbar sind.
`SliderUsage`	UsageMode [▶ 2836]	Bestimmt, ob die einzelnen Farbwerte als Slider angezeigt werden und ob diese bearbeitbar sind.
`SliderValueUpdate`	`DataUpdate`	Legt fest, ob das Farbsymbol, während `onChanged` oder am Ende `onConfirmed` einer Interaktion mit den Slidern aktualisiert werden soll.

Kategorie: Appearance

Die folgenden Attribute bestimmen das Aussehen [▶ 2817] des Controls.

Attribut	Typ	Beschreibung
`Orientation`	Orientation [▶ 2836]	Orientierung des Controls; analog zu der Orientierung der LinearGauge-Controls. `Horizontal` oder `Vertical`.
`BoxSize`	`Number (px)`	Bestimmt die Breite oder Höhe der Farbbox je nach Orientierung (Wert von Orientation).
`ValuesSize`	`Number (px)`	Bestimmt die Breite oder Höhe der angezeigten Farbwerte je nach Orientierung (Wert von Orientation).

Themed Resources

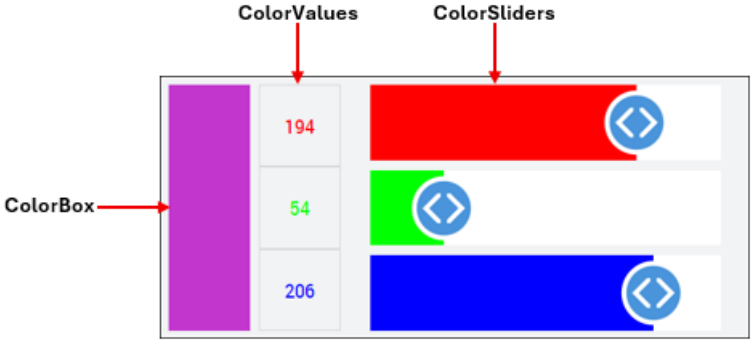
Mit folgenden Attributen kann das Aussehen von Unter-Controls (den Textboxen für die Farbwerte) unter den Themes Einstellungen verändert werden. Eine detaillierte Beschreibung der einzelnen Attribute ist in der Standard-HMI-Dokumentation zu finden.

Attribut	Typ
`TextFontFamily`	`FontFamily`
`TextFontSize`	`MeasurementValue`
`TextFontSizeUnit`	`MeasurementValue`
`TextFontStyle`	`FontStyle`
`TextFontWeight`	`FontWeight`
`TextPadding`	`Padding`

7.2.2.1 Anzeigeelemente

Elemente

Die folgenden Elemente können in dem Farbcontrol angezeigt werden. Die entsprechenden Verwendungsattribute (`BoxUsage, ValuesUsage und SliderUsage`) legen jeweils fest, ob das Element versteckt (`Hide`) oder angezeigt (`ViewOnly`) wird und ob ein Nutzer mit Hilfe dieses Elements die Farbe verändern kann (`Editable`).



Name	Position	Beschreibung
ColorBox	Links oder oben	Dies ist ein rechteckiger Bereich, in dem die aktuelle Farbe angezeigt wird. Wenn die Eigenschaft `BoxUsage` auf „Editable“ eingestellt ist, öffnet ein Klick auf die `ColorBox` ein browserspezifisches Dialogfeld zur Auswahl der Farbe. Je nach Browser kann dies wie ein separates Fenster oder wie ein Overlay aussehen oder es wird nichts angezeigt. Da jeder Browser das Aussehen und die Funktionalität des Dialogs selbst steuert, haben wir darauf keinen Einfluss. Zum Beispiel macht die Farbpipette einen Screenshot des Bildschirms, so dass sie nicht mit wechselnden Bildern funktioniert. Weitere Informationen dazu finden Sie im Kapitel Farbboxen in verschiedenen Browsern [▶ 2818].
ColorValues	Mitte	Dieser Bereich umfasst Textfelder zur Anzeige und Bearbeitung der im Control gespeicherten Farbwerte. Die Anzahl der Wertelemente und ihre Textfarbe werden aus der Eigenschaft **ColorChannels** übernommen.
ColorSliders	Rechts oder unten	Dieser Bereich umfasst lineare Schieberegler, die den **ColorValues** entsprechen.

Anordnung

Die Anordnung der Anzeigeelemente hängt von dem `Orientation` Attribut ab. Bei „Horizontal“ werden die Elemente nebeneinander platziert und bei „Vertical“ werden sie übereinander platziert. Je nach Orientierung legen die Attribute `BoxSize` und `ValuesSize` die Breite oder Höhe der jeweiligen Elemente fest. Die Breite oder Höhe der Slider entspricht automatisch dem Rest des verfügbaren Platzes.

HINWEIS

Kleine Größen

Wenn das Control in der gewählten Ausrichtung sehr klein ist, kann es sein, dass nicht alle Elemente vollständig und korrekt angezeigt werden können. Dies betrifft vor allem die Schieberegler. Ab einer bestimmten Größe wird in der JavaScript-Konsole eine Warnung ausgegeben, dass die Größe die Funktion des Controls beeinträchtigen kann.

Anzahl

Die Anzahl der angezeigten Farbwerte und Slider entspricht immer der Anzahl der Kanäle im eingestellten Farbformat [▶ 2822]. Das Farbformat wird mittels des `ColorChannels` Attributs festgelegt.

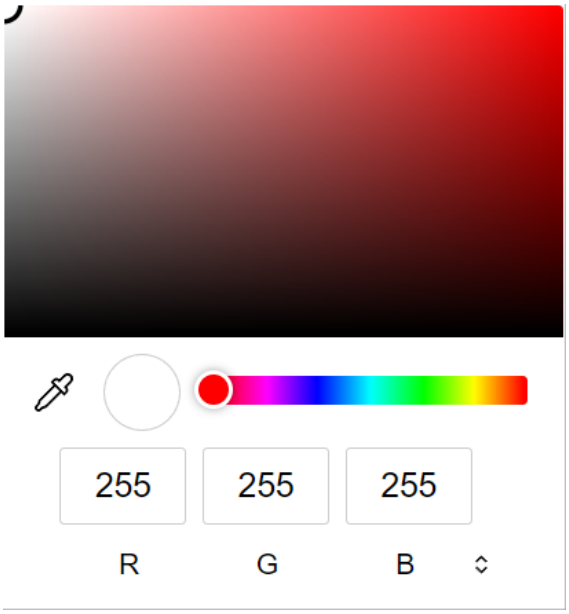
Text

Die Textboxen für die Farbwerte können über verschiedene Schrifteigenschaften mit Hilfe von Themed Resources im Theme-Editor angepasst werden.

7.2.2.1.1 Farbboxen in verschiedenen Browsern

Wenn das `BoxUsage` Attribute auf `Editable` gestellt ist, kann mit einem Klick auf die Farbbox ein weiterer Auswahl-Dialog für die Farbe geöffnet werden. Dieser Dialog ist Browser-abhängig. Folgend sind einige Dialoge der meistbenutzten Browser aufgelistet:

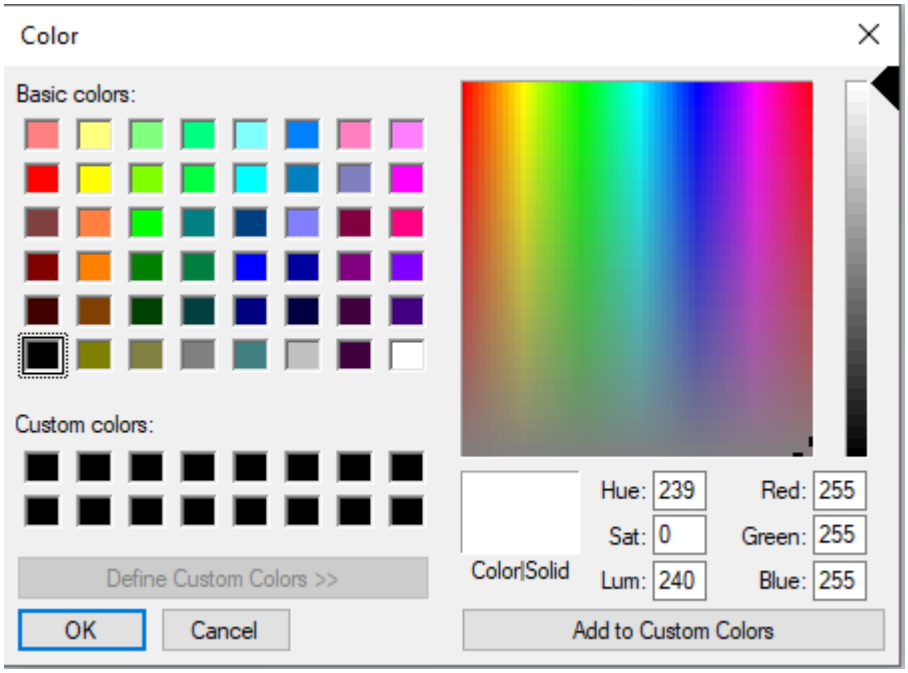
Google Chrome und Microsoft Edge



Pipette

Bei der Pipetten-Funktion in Chromium-basierten Browsern wird bei Auswahl des Tools ein Screenshot des Browser-Fensters gemacht. Die selektierte Farbe stammt dann von diesem Screenshot. Dies bedeutet, dass bei sich ändernden Bildern die ausgewählte Farbe nicht die ist, die zum Zeitpunkt des Klicks angezeigt wird. Daher wird empfohlen, vor der Nutzung der Pipetten-Funktion den Bildlauf mittels des `ImageFreeze` Attributs anzuhalten.

Mozilla Firefox



7.2.2.2 Farbdaten

Das Attribut `ColorSymbol` kann an verschiedene Symbole gebunden werden. Um Unstimmigkeiten zwischen dem ausgegebenen Farbwert und dem Datentyp des Symbols zu vermeiden, liest das Color-Control den Typ des Symbols und versucht, automatisch den richtigen Rückgabetyp abzuleiten.

Wenn zum Beispiel drei Farbkanäle angezeigt werden sollen, in der SPS jedoch ein Array vom Typ `TcVnVector4_LREAL` verknüpft wird, stimmen diese Typen nicht überein. Dies erkennt das Color Control automatisch und schreibt ein Array mit vier anstatt mit drei Werten in das Symbol, sodass kein Fehler auftritt. Der übrige Kanal wird dabei leer gelassen.

Das schreibgeschützte Attribut `ColorValue` dient dazu, die Farbe programmatisch zugänglich zu machen, ohne ein Symbol an die Eigenschaft `ColorSymbol` binden zu müssen. Ansonsten gibt es keinen Unterschied. Eine gleichzeitige Nutzung beider Varianten ist möglich. Hierbei ist zu beachten, dass `ColorValue` ebenfalls den Datentypen des Symbols ausgibt, welches an `ColorSymbol` gebunden ist.

Mit dem Attribut `SliderValueUpdate` wird festgelegt, ob das Farbsymbol, während (`onChanged`) oder am Ende einer Interaktion (`onConfirmed`) mit den Slidern aktualisiert werden soll. Wenn der Farbwert während einer Sliderbewegung (`onChanged`) geändert wird und ein Symbol an das Attribut `ColorSymbol` gebunden ist, kann es vorkommen, dass der Wert nach dem Loslassen des Sliders auf einen älteren Wert zurückspringt. Wenn dies Verhalten nicht gewünscht ist, sollte `SliderValueUpdate` auf `onConfirmed` gesetzt werden.

In Analogie zur Eigenschaft `ColorValue` bietet die Methode `SetColorValue` eine Möglichkeit, die Farbwerte dieses Controls einzustellen, ohne ein Symbol mit der Eigenschaft `ColorSymbol` zu verknüpfen.

Die beiden Zugriffsarten auf den Farbwert sind anhand der folgenden Beispiele erläutert:

- Verknüpfung mit PLC [▶ 2820]
- Programmatische Nutzung [▶ 2821]

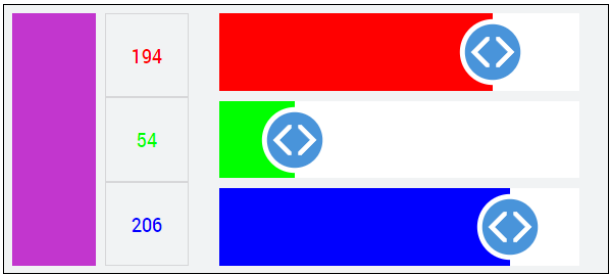
Wie die Farbdaten auch für andere Anwendungsfälle verwendet werden können zeigt das Beispiel Konvertierung zu HMI-Farbobjekt [▶ 2821].

7.2.2.2.1 Verknüpfung mit PLC

Im einfachsten Fall kann das Color Control genutzt werden, um eine Farbvariable aus der PLC zu repräsentieren. Dazu wird zum Beispiel eine PLC-Variable vom Typ `TcVnVector4_LREAL` mit dem `ColorSymbol` Attribut verknüpft. Des Weiteren werden alle `Vector3` und `Vector4` Datentypen sowie `INT, DINT, SINT, UINT, USINT` und `LREAL` unterstützt.

Nun werden PLC-Variable und das Control miteinander synchronisiert:

Expression	Type	Value
aColor	TcVnVector4_LREAL	
aColor[0]	LREAL	194
aColor[1]	LREAL	54
aColor[2]	LREAL	206
aColor[3]	LREAL	0



Das vierte Element des PLC-Arrays wird ignoriert, da in dem Control standardmäßig nur drei Kanäle als RGB angezeigt werden. Dies kann über das Farbformat [▶ 2822] eingestellt werden.

7.2.2.2.2 Programmatische Nutzung

Wenn der Farbwert des Color Controls nicht direkt mit einem Symbol verknüpft werden soll, kann stattdessen programmatisch darauf zugegriffen werden. Um eine Farbe in dem Control einzustellen, verwenden Sie die Funktion `setColorValue`, entweder über JavaScript-Code oder über den **Actions and conditions** Dialog:

```
TcHmi.Controls.get("Background").setColorValue([0, 0, 255])
```

Um die Farbe des Controls abzufragen, verwenden Sie das ReadOnly Attribut `ColorValue`:

In JavaScript muss das Attribut mittels der `getColorValue` Methode abgerufen werden:

```
const color = TcHmi.Controls.get("Background").getColorValue()
```

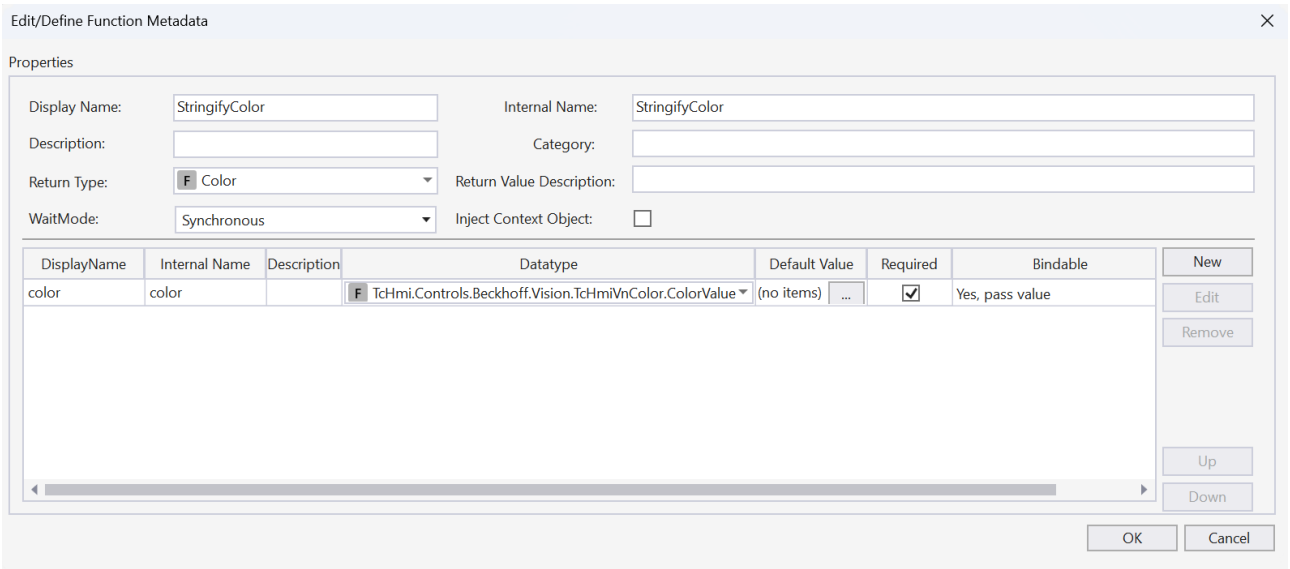
7.2.2.2.3 Konvertierung zu HMI-Farbobjekt

Um eine Farbe in der TwinCAT HMI z. B. als Hintergrund oder für andere Eigenschaften wie Text oder Rahmen nutzen zu können, muss diese in ein Objekt vom Typ TcHmi.Color konvertiert werden. Das folgende Beispiel zeigt, wie eine solche Funktion angelegt wird und welche Schnittstellen diese haben muss.

Die allgemeine Verwendung von Funktionen wird in der HMI-Dokumentation erklärt.

Nach dem Erstellen der TypeScript Funktion muss ein Parameter angelegt werden. Dieser bekommt den Farbwert z. B. von der PixelColor-Eigenschaft des ImageControls oder den ColorValue vom Color Control übergeben und muss daher vom Typ `Array` angelegt werden. Es ist zu beachten, dass sich der übergebene Parameter zum Zeitpunkt der Funktionsausführung aus Timing-Gründen vom Wert der entsprechenden Control-Eigenschaften unterscheiden kann.

Die Funktion muss einen Wert vom Typ `Color` zurückgeben, der dann für andere Eigenschaften im HMI verwendet werden kann.



Folgendes Codebeispiel zeigt die Konvertierung eines RGB-Farbwerts:

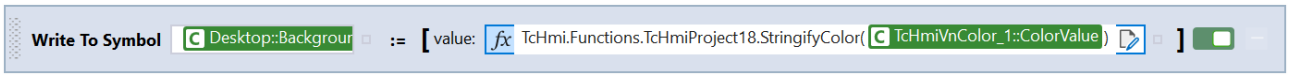
```
namespace TcHmi {
    export namespace Functions {
        export namespace TcHmiProject {
            export function StringifyColor(color: any){
```

```
                if (color.length >= 3){
                    return { color: `rgb(${color[0]}, ${color[1]}, ${color[2]})` };
                }
                throw new Error("Could not stringify color.");
            }
        }
    }
}
TcHmi.Functions.registerFunctionEx('StringifyColor', 'TcHmi.Functions.TcHmiProject',
TcHmi.Functions.TcHmiProject.StringifyColor);
```

Um als Farbobjekt akzeptiert zu werden, muss der Rückgabewert den Typ `Color` haben und wie folgt aufgebaut sein:

```
{
    color: 'rgb(<red>, <green>, <blue>)'
}
```

Eine solche Funktion kann z.B. genutzt werden, indem der Farbwert eines TcHmiVnColor Controls mit der Funktion umgewandelt und dann als Hintergrundfarbe eines anderen UI-Elements festgelegt wird.



7.2.2.3 Farbformat

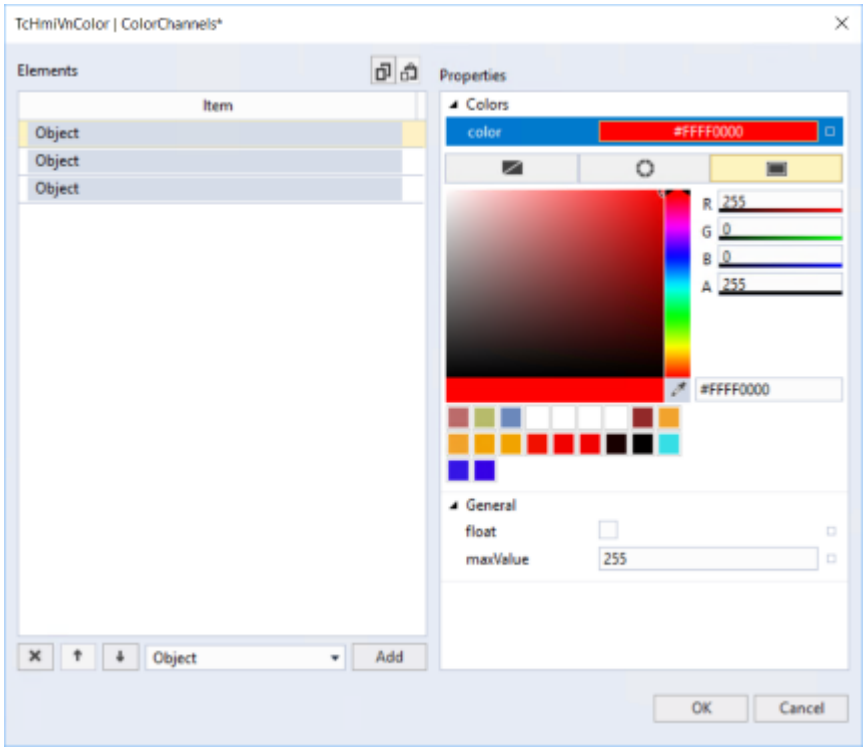
Es gibt viele verschiedene Möglichkeiten, eine Farbe darzustellen. Auf die gebräuchlichsten Formate wie Mono (1 Kanal Grauwert), RGB Farbraum (3 Kanäle Farbwert), HSV / HLS Farbraum (3 Kanäle Farbwert) wird im Folgenden genauer eingegangen. Das Control bietet jedoch viele Konfigurationsmöglichkeiten, so dass auch spezielle Farbformate eingestellt und verwendet werden können.

Einstellung des Farbformats

Zur Anpassung des Controls an beliebige Farbformate, können die Rahmenbedingungen im Detail eingestellt werden:

- Anzahl der Kanäle (Objekte)
- Maximalwerte jedes Kanals
- Datentyp (Float oder Integer) jedes Kanals
- Anzeigefarbe jedes Kanals. Diese wird als Farbe des jeweiligen Sliders und des angezeigten Farbwerts benutzt.

Die Anpassungen können über den Dialog des Attributs `ColorChannels` vorgenommen werden. Als Standard ist das RGB Format voreingestellt. Um z. B. ein Mono 8 Format einzustellen, müssen zunächst zwei der drei vorhandenen Kanäle (Object) gelöscht werden. Danach kann die Farbdarstellung auf einen mittleren Grauwert wie 128 in den R, G und B Kanälen angepasst werden.



Es sind Konfigurationsbeispiele zu folgenden Farbformaten verfügbar:

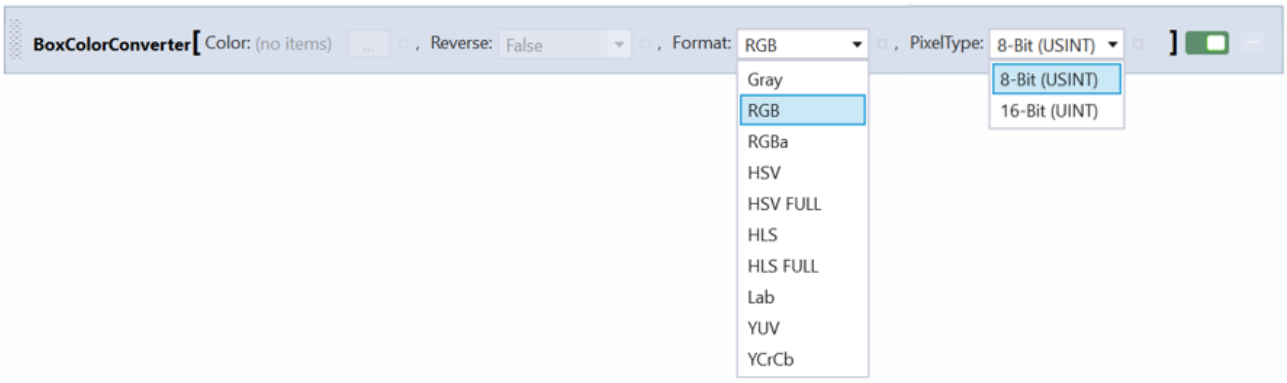
- RGB (Standard) [▶ 2824]
- HSV / HLS (Wertanpassungen) [▶ 2825]
- Prozentuales RGB (Benutzerdefiniertes Format) [▶ 2826]

Farbbox-Konvertierung

Dieser Abschnitt ist nur relevant, wenn die Farb-Box verwendet wird und die Farbe nicht dem Standard-RGB-Format entspricht.

Für die Anzeige des Farbwerts mittels Zahlen und Slidern wird kein tieferes Wissen über das Farbformat benötigt, da es sich lediglich um die Darstellung von Zahlenwerten handelt. Um die Farbe jedoch in der Farb-Box anzeigen zu können, muss definiert sein, wie die einzelnen Farbkanäle und deren Werte zu interpretieren sind. Zum Beispiel müssen RGB-Werte anders dargestellt werden als HSV-Werte.

Da die browserspezifischen Farbeingabeelemente standardmäßig RGB-Farben verarbeiten, müssen alle anderen Farbformate zunächst in RGB umgewandelt werden, damit eine Anzeige in der Farb-Box möglich ist. Diese Konvertierung ist über das Attribut `BoxColorConversion` einzustellen. Für die Formate Gray, RGBa, HSV, HSV FULL, HLS, HLS FULL, Luv, YUV und YCrCb, sowie für 8- und 16-Bit Datentypen existieren bereits Konvertierungsfunktionen.



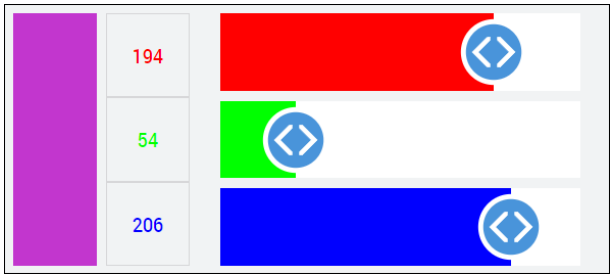
Allgemeiner Farbkonverter

Zusätzlich zu dieser speziellen Konvertierungsfunktion für die Farb-Box ist auch ein allgemeiner Farbkonverter mit demselben Satz von Farbformaten zur allgemeinen Verwendung in der HMI als Funktion ConvertColor [▶ 2833] verfügbar.

Für gänzlich andere Farbformate oder Farbformate mit anderen Skalierungen können benutzerdefinierte Funktionen [▶ 2826] erstellt werden.

7.2.2.3.1 Standard RGB(a)-Farbformat

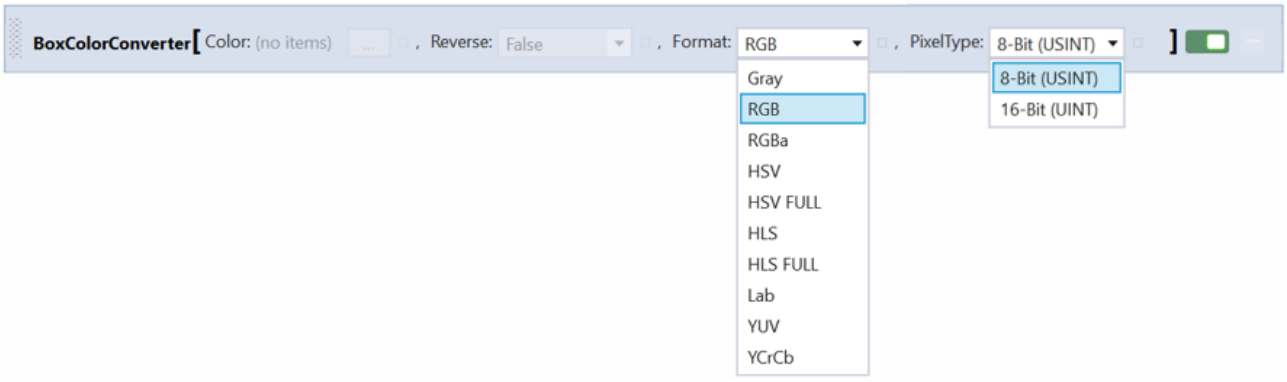
Standardmäßig sind drei Kanäle definiert, die alle als Integer mit einem Maximalwert von 255 konfiguriert sind und deren Anzeige-Farben dem RGB-Farbformat entsprechen.



Die Werte des `ColorChannels` Attributs können per Skript wie folgt geschrieben werden:

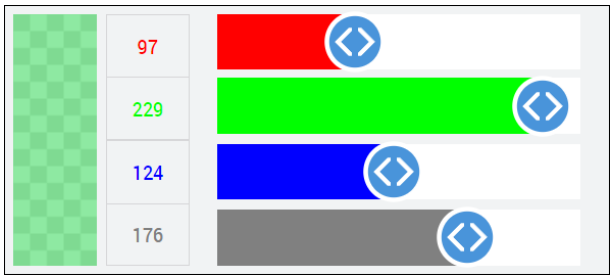
```
[
  {
    "color": {
      "color": "rgba(255, 0, 0, 1)"
    },
    "float": false,
    "maxValue": 255.0
  },
  {
    "color": {
      "color": "rgba(0, 255, 0, 1)"
    },
    "float": false,
    "maxValue": 255.0
  },
  {
    "color": {
      "color": "rgba(0, 0, 255, 1)"
    },
    "float": false,
    "maxValue": 255.0
  }
]
```

Das `BoxColorConversion` Attribut ist auf die mitgelieferte Funktion `BoxColorConverter` gesetzt und diese ist mit dem Format „RGB“ und dem Pixeltype „8-Bit (USINT)“ konfiguriert.



RGBa-Farbformat

Um auch transparente Farben darstellen zu können, muss ein weiterer Farbkanal im Attribut `ColorChannels` hinzugefügt werden. Es bietet sich an, diesen mit der Anzeigefarbe „rgba(128, 128, 128, 1)“ zu konfigurieren. Da der `BoxColorConverter` schon über eine Konvertierung zu RGBa verfügt, muss diese anschließend als Farbformat ausgewählt werden.

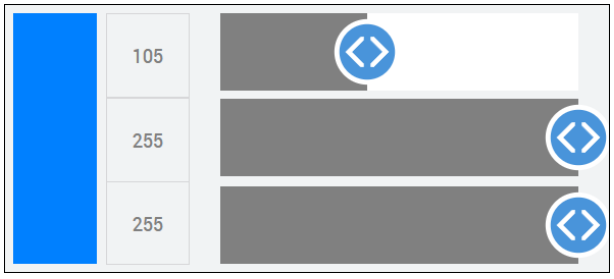


7.2.2.3.2 HSV-/ HLS (FULL)-Farbformat

Wenn eine Farbe im HSV-Format mit dem Color Control dargestellt werden soll, sind einige Anpassungen notwendig. Der Wertebereich des „Hue“ Kanal (engl. für „Farbwert“) ist nur von 0 bis 180 definiert, weswegen der `maxValue` des ersten Kanals auf 180 herabgesetzt werden muss. Die beiden weiteren Kanäle sind auf 0 bis 255 skaliert. Da die einzelnen Kanäle des HSV-Formats nicht spezifischen Farben zuzuordnen sind, sollten die Farben für die Darstellung der Slider und Zahlenwerte z.B. auf einen einheitlichen Grauwert gestellt werden, sodass keine Verwirrung entsteht.

Für den HLS-Farbraum können die gleichen Einstellungen und Wertebereiche verwendet werden, da sich lediglich die Berechnung der L und S Kanäle unterscheidet.

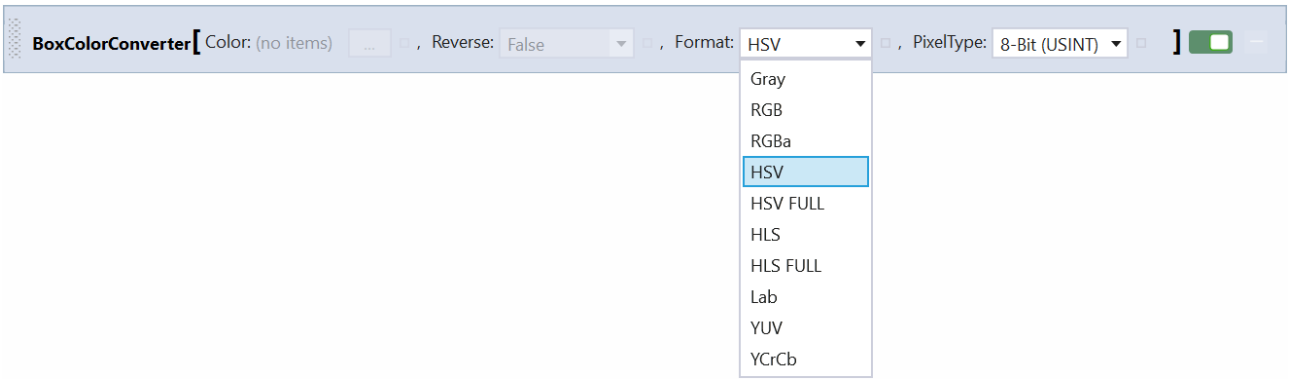
Bei der Verwendung der Farbformate „HSV FULL“ und „HLS FULL“ sind alle Farbkanäle von 0 bis 255 definiert.



Die Werte des `ColorChannels` Attributs können per Skript wie folgt geschrieben werden:

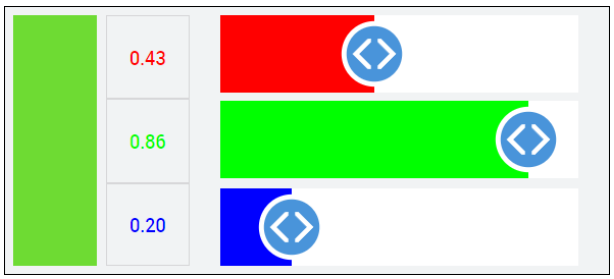
```
[
  {
    "color": {
      "color": "rgba(90, 90, 90, 1)"
    },
    "float": false,
    "maxValue": 180.0
  },
  {
    "color": {
      "color": "rgba(90, 90, 90, 1)"
    },
    "float": false,
    "maxValue": 255.0
  },
  {
    "color": {
      "color": "rgba(90, 90, 90, 1)"
    },
    "float": false,
    "maxValue": 255.0
  }
]
```

Schließlich muss die Konvertierung für die Farb-Box in dem `BoxColorConversion` Attribut angepasst werden. Dazu muss das entsprechende Farbformat in der `BoxColorConverter` Funktion ausgewählt werden:



7.2.2.3.3 Benutzerdefiniertes Farbformat

Als weiteres Beispiel der Color Control Konfiguration wird ein RGB-Farbformat mit einem Wertebereich zwischen 0 und 1 betrachtet. Um dies in dem Color Control darzustellen, müssen die Kanäle im `ColorChannel` Attribut als `float` und mit einem `maxValue` von `1` eingestellt werden.



Die Werte des `ColorChannels` Attributs können per Skript wie folgt geschrieben werden:

```
[
  {
    "color": {
      "color": "rgba(255, 0, 0, 1)"
    },
    "float": true,
    "maxValue": 1.0
  },
  {
    "color": {
      "color": "rgba(0, 255, 0, 1)"
    },
    "float": true,
    "maxValue": 1.0
  },
  {
    "color": {
      "color": "rgba(0, 0, 255, 1)"
    },
    "float": true,
    "maxValue": 1.0
  }
]
```

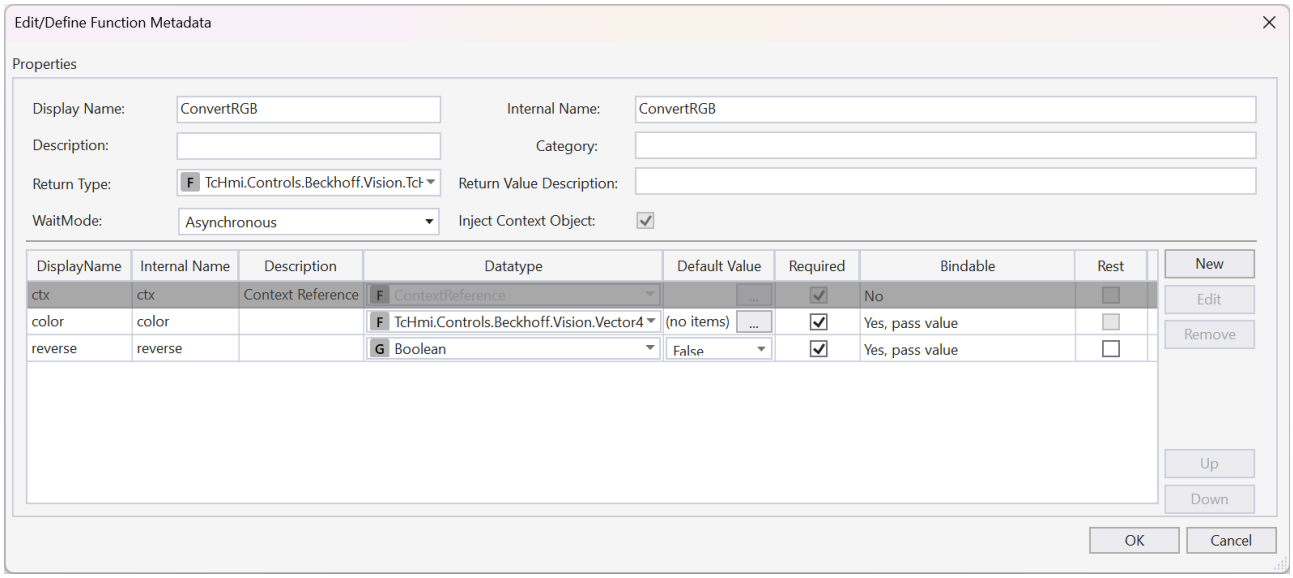
ColorBoxFormatting Funktion

Damit die Farb-Box trotz des veränderten Wertebereich die richtige Farbe anzeigt, muss das `BoxColorConversion` Attribut angepasst werden. Anders als bei den Standard-Konvertierungen existiert hierfür keine vorgefertigte Funktion. Stattdessen ist die Erstellung einer eigenen Funktion notwendig. Das folgende Beispiel zeigt, wie eine solche Funktion für den oben genannten Anwendungsfall angelegt wird und welche Schnittstellen diese haben muss.

Die allgemeine Verwendung von Funktionen wird in der HMI-Dokumentation erklärt.

Nach dem Erstellen der TypeScript Funktion muss der WaitMode auf Asynchronous umgestellt werden. Das hat zur Folge, dass der Parameter „ctx“ angelegt wird, der für die Rückgabe notwendig ist. Danach müssen zwei Parameter angelegt werden. Der erste bekommt den zu konvertierenden Farbwert übergeben und muss daher vom Typ `TcHmi.Controls.Beckhoff.Vision.Vector4` angelegt werden. Der zweite Parameter gibt die Konvertierungsrichtung an und muss als `Boolean` angelegt werden. Mit `True` werden die Werte nach RGB konvertiert und mit `False` von RGB zurück in das benutzerdefinierte Farbformat. Es ist zu beachten, dass sich diese übergebenen Parameter zum Zeitpunkt der Funktionsausführung aus Timing-Gründen von den Werten der entsprechenden Control-Eigenschaften unterscheiden können.

Die Funktion muss einen Wert vom Typ `TcHmi.Controls.Beckhoff.Vision.TcHmiVnColor.ColorValue` zurückgeben, der dann in der Farb-Box angezeigt wird.



Im folgenden Codebeispiel wird die Skalierung der Farbwerte angepasst. Wenn der Parameter `reverse = false` ist, werden die Farbwerte vom Control im benutzerdefinierten Format an die Funktion übergeben. Das Farbformat muss dann in RGB(a) konvertiert werden, um in der ColorBox angezeigt werden zu können. Die übergebenen Farbwerte im Parameter „color“ sind immer vierkanalig. Wenn keine transparente Farbe in der ColorBox angezeigt werden soll, müssen die zurückgegebenen Farbwerte dreikanalig sein. Alternativ kann auch der vierte Kanal auf 255 gesetzt werden.

Wenn der Parameter `reverse = true` ist, werden Farbwerte vom Farbauswahl-Dialog im Format RGBa an die Funktion übergeben. Das Farbformat muss dann in das benutzerdefinierte Farbformat konvertiert werden. Der vierte Kanal wird aus den Farbwerten des Controls übernommen, weil der Farbauswahl-Dialog keine Transparenz unterstützt.

```
namespace TcHmi {
    export namespace Functions {
        export namespace TcHmiProject {
            export function ConvertRGB(ctx: any, color: any, reverse: any) {
                // Important: color has 4 channels. If the ColorBox should be RGB, return a 3
channel color or set the fourth channel (a) to 255
                if (!reverse) {
                    ctx.success(color.slice(0, 3).map((value: number) => value * 255));
                }
                else {
                    ctx.success(color.slice(0, 3).map((value: number) => value / 255));
                }
            }
        }
    }
}
TcHmi.Functions.registerFunctionEx('ConvertRGB', 'TcHmi.Functions.TcHmiProject',
TcHmi.Functions.TcHmiProject.ConvertRGB);
```

Damit ist gewährleistet, dass die Farb-Box die richtige Farbe anzeigt und dass ein manuelles Ändern der Farbe in der Farb-Box zu korrekten Farbwerten führt.

7.2.3 Rectangle (Interactive)

Das Rectangle (Interactive) Control bietet eine Umgebung, um ein Rechteck anzeigen oder per Klicks zeichnen zu können. Hauptsächlich wird das Control innerhalb des Image Watch [▶ 2790]-Controls verwendet; es kann aber auch alleinstehend genutzt werden. Somit können z.B. Bildregionen ausgewählt werden, um Objekte zu markieren oder ROIs festzulegen. Neben einem Rechteck können auch Quadrate, Kreise oder Ellipsen gezeichnet werden. Dazu müssen die Attribute StrokeShape und AspectRatio entsprechend angepasst werden.

Für alle folgenden Attribute besitzt das Control eine Getter- und eine Setter-Methode. Alle weiteren Attribute, Events und Permissions die die Basisklasse **Controls** bereitstellt, finden Sie in der TwinCAT HMI Dokumentation.

Kategorie: Colors

Attribut	Typ	Beschreibung
`StrokeColor`	`SolidColor`	Farbe der Konturen des gezeichneten Rechtecks.
`HandleColor`	`SolidColor`	Farbe der Griffe des gezeichneten Rechtecks.

Kategorie: Common

Attribut	Typ	Beschreibung
`ShapeData`	`TcVnRotatedRectangle`	Enthält die Beschreibung des Rechtecks. Kann sowohl zum Abrufen eines gezeichneten Rechtecks genutzt werden als auch zum Einstellen eines spezifischen Rechtecks.
`AspectRatio`	`Number`	Gibt das Seitenverhältnis an, dass beim Zeichnen des Rechtecks erzwungen wird. Beim Wert `0` wird kein bestimmtes Seitenverhältnis erzwungen.

Funktion	Beschreibung
`clear`	Löscht das gezeichnete Rechteck und die Beschreibung aus dem `ShapeData`-Attribut.

Kategorie: Usage

Attribut	Typ	Beschreibung
`InitialSelection`	`InitialSelection`	Legt fest, ob das Rechteck mit zwei Klicks oder mit einer streichenden Bewegung oder mittels beider Optionen gezeichnet werden kann.
`CenterHandle`	`Boolean`	Zeigt ein Element in der Mitte des Rechtecks an, um dieses zu verschieben.
`CornerHandles`	`Boolean`	Zeigt Elemente an den Ecken des Rechtecks an, um dessen Größe zu verändern.
`EdgeHandles`	`Boolean`	Zeigt Elemente an den Kanten des Rechtecks an, um dessen Größe zu verändern.
`RotationHandle`	`Boolean`	Zeigt ein Element außen am Rechteck an, um dieses zu drehen.
`ShapeDragging`	`Boolean`	Legt fest, ob das Rechteck an den Kanten verschoben werden kann.
`AngleInterval`	`Number`	Gibt an, in welchen Schritten das Rechteck gedreht werden kann. Bei `0` kann das Rechteck beliebig gedreht werden.
`Boundary`	`Boundary`	Legt über die fünf Eigenschaften `X, Y, Width, Height` und `IsActive` einen Bereich fest, in dem die Form gezeichnet und bearbeitet werden kann. Die Form kann diesen Bereich nicht verlassen.
`DataUpdate`	`DataUpdate`	Legt fest, ob `ShapeData` während `onChanged` oder am Ende `onConfirmed` einer grafischen Interaktion aktualisiert werden soll.

Kategorie: Appearance

Attribut	Typ	Beschreibung
StrokeThickness	MeasurementValue	Linien-Dicke des Rechtecks.
HandleSize	MeasurementValue	Größe der Elemente zum Verschieben und Verändern des Rechtecks.
StrokeShape	StrokeShape	Legt fest, ob ein Rechteck oder eine Ellipse gemalt wird. Die Beschreibung ist für beides identisch. Um ein Quadrat oder einen Kreis zu erhalten, kann der AspectRatio auf 1 gesetzt werden.
ClickableSize	MeasurementValue	Größe der klickbaren Fläche um o.g. Elemente zum Verschieben und Verändern des Rechtecks. Hiermit kann die Benutzung per Touchscreen vereinfacht werden, ohne das Aussehen zu verändern.
Offset	Vector2	Legt fest, mit welcher Verschiebung die Form angezeigt werden soll.
Scale	Number	Legt fest, in welcher Skalierung die Form angezeigt werden soll. Der Wert 1 entspricht dabei 100%.

Events

Event	Beschreibung
.onShapeChanged	Dieses Event wird ausgelöst, sobald sich das Attribut ShapeData ändert.
.onShapeConfirmed	Dieses Event wird ausgelöst, wenn eine Form gesetzt oder eine Interaktion mit dieser abgeschlossen wurde.

7.2.4 Polygon (Interactive)

Das Polygon (Interactive) Control bietet eine Umgebung, um ein Polygon anzeigen oder per Klicks zeichnen zu können. Hauptsächlich wird das Control innerhalb des Image Watch [▶ 2790]-Controls verwendet; es kann aber auch alleinstehend genutzt werden. Neben einem Polygon können z. B. auch Linien oder einzelne Punkte angezeigt oder gezeichnet werden. Dazu muss die NumPoints Eigenschaft entsprechend auf 1 oder 2 gesetzt werden.

Für alle folgenden Attribute besitzt das Control eine Getter- und eine Setter-Methode. Alle weiteren Attribute, Events und Permissions die die Basisklasse **Controls** bereitstellt, finden Sie in der TwinCAT HMI Dokumentation.

Kategorie: Colors

Attribut	Typ	Beschreibung
StrokeColor	SolidColor	Farbe der Konturen des gezeichneten Polygons.
HandleColor	SolidColor	Farbe der Griffe des gezeichneten Polygons.

Kategorie: Common

Attribut	Typ	Beschreibung
ShapeData	PolygonData	Enthält die Beschreibung des Polygons. Kann sowohl zum Abrufen eines gezeichneten Polygons genutzt werden als auch zum Einstellen eines spezifischen Polygons.
NumPoints	Number	Gibt an, wie viele Punkte das vollständige Polygon haben soll. Bei 0 sind beliebig viele Punkte möglich. Diese Angabe ist wichtig, um das Polygon als vollständig handhaben zu können.
Closed	Boolean	Gibt an, ob das Polygon geschlossen ist oder ob es sich um eine Polylinie handelt.

Funktion	Beschreibung
clear	Löscht das gemalte Polygon und die Beschreibung aus dem ShapeData-Attribut.

Kategorie: Usage

Attribut	Typ	Beschreibung
RotationHandle	Boolean	Zeigt ein Element außen am Polygon an, um dieses zu drehen.
AngleInterval	Number	Gibt an, in welchen Schritten das Polygon gedreht werden kann. Bei 0 kann das Polygon beliebig gedreht werden.
Boundary	Boundary	Legt über die fünf Eigenschaften X, Y, Width, Height und IsActive einen Bereich fest, in dem die Form gezeichnet und bearbeitet werden kann. Die Form kann diesen Bereich nicht verlassen.
DataUpdate	DataUpdate	Legt fest, ob ShapeData während onChanged oder am Ende onConfirmed einer grafischen Interaktion aktualisiert werden soll.

Kategorie: Appearance

Attribut	Typ	Beschreibung
`StrokeThickness`	`MeasurementValue`	Linien-Dicke des Polygons.
`HandleSize`	`MeasurementValue`	Größe der Elemente zum Verschieben und Verändern des Polygons.
`ClickableSize`	`MeasurementValue`	Größe der klickbaren Fläche um o.g. Elemente zum Verschieben und Verändern des Polygons. Hiermit kann die Benutzung per Touchscreen vereinfacht werden, ohne das Aussehen zu verändern.
`Offset`	`Vector2`	Legt fest, mit welcher Verschiebung die Form angezeigt werden soll.
`Scale`	`Number`	Legt fest, in welcher Skalierung die Form angezeigt werden soll. Der Wert `1` entspricht dabei 100%.

Events

Event	Beschreibung
`.onShapeChanged`	Dieses Event wird ausgelöst, sobald sich das Attribut `ShapeData` ändert.
`.onShapeConfirmed`	Dieses Event wird ausgelöst, wenn eine Form gesetzt oder eine Interaktion mit dieser abgeschlossen wurde.

7.2.5 Hilfsfunktionen

Folgende Hilfsfunktionen werden als Standardverhalten von bestimmten Controls genutzt oder um Interoperabilität mit PLC-Datentypen zu gewährleisten.

BoxColorConverter

Die Funktion `BoxColorConverter` ist in der Kategorie `Vision` gelistet. Sie wird als Standard-Wert für das `BoxColorConversion` Attribut des Color Controls [▶ 2815] verwendet. Sie konvertiert zwischen dem RGB und dem durch den `Format`-Parameter angegebenen Farbformat.

Parameter	Datentyp	Beschreibung
`ctx`	`SelectableRequired<TcHmi.Context, 'success' | 'error'> | null`	Wird HMI-intern verwendet, um asynchrone Rückgaben zu empfangen. Kann bei Aufrufen in Javascript aber auf `null` gesetzt werden.
`Color`	`Vector4`	Der zu konvertierende Farbwert.
`Reverse`	`Boolean`	Richtung der Konvertierung: `false`: `format` zu RGBa `true`: RGBa zu `format`
`Format`	`ColorSpace`	Format für Konvertierung: `Gray, RGB, HSV, HLS, RGBa, HSV FULL, HLS FULL, Lab, YUV, YCrCb`

`PixelType`	`ElementType`	Skalierung der Farbwerte:
		`8-Bit (USINT),`
		`16-Bit (UINT)`

Da die Funktion asynchron arbeitet, gibt sie die konvertierte Farbe als zukünftigen Wert, also als `Promise<ColorValue | null>` zurück. Siehe `ConvertColor` zur Handhabung von Promise. Fehler bei der Konvertierung führen dazu, dass `null` zurückgegeben wird.

ConvertColor

Die Funktion `ConvertColor` ist in der Kategorie `Vision` gelistet. Sie konvertiert ein Farbarray zwischen den Formaten Gray, RGB, RGBa, HSV, HSV FULL, und HLS, HLS FULL, Lab, YUV und YCrCb.

Parameter	Datentyp	Beschreibung
`ctx`	`SelectableRequired<TcHmi.Context, 'success' | 'error'> | null`	Wird HMI-intern verwendet, um asynchrone Rückgaben zu empfangen. Kann bei Aufrufen in Javascript aber auf `null` gesetzt werden.
`Color`	`Vector4`	Der zu konvertierende Farbwert.
`SrcFormat`	`ColorSpace`	Format des eingehenden Farbwerts: `Gray, RGB, HSV, HLS, RGBa, HSV FULL, HLS FULL, Lab, YUV, YCrCb`
`DestFormat`	`ColorSpace`	Format, in das konvertiert wird: `Gray, RGB, HSV, HLS, RGBa, HSV FULL, HLS FULL, Lab, YUV, YCrCb`
`Precision`	`Precision`	Legt fest, auf wie viele Dezimalstellen gerundet werden soll. Bei `Unrounded` wird nicht gerundet.
`SrcPixelType`	`ElementType`	Skalierung der eingehenden Farbwerte: `8-Bit (USINT), 16-Bit (UINT)`
`DestPixelType`	`ElementType`	Skalierung der konvertierten Farbwerte: `8-Bit (USINT), 16-Bit (UINT)`

Fehler bei der Konvertierung führen dazu, dass null zurückgegeben wird. Da die Funktion asynchron ist, gibt sie die konvertierte Farbe als zukünftigen Wert, also als `Promise<ColorValue | null>` zurück.

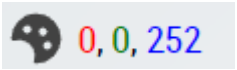
Um z.B. die Farbe Rot im RGB-Format in das HLS-Format zu konvertieren, kann folgender Funktionsaufruf verwendet werden:

```
let rgb = [255, 0, 0];
let promise = TcHmi.Functions.Beckhoff.Vision.ConvertColor(null, rgb, "RGB", "HLS", "2 Decimals",
"8-Bit (USINT)", "8-Bit (USINT)");
promise.then(hls => console.log(hls.toString()));

//console output
//0,127.5,255
//console output with DestPixelType "16-Bit (UINT)"
//0,32640,65280
```

Um den zukünftigen `ColorValue` weiterverarbeiten zu können, kann dieser mit `await` abgewartet werden. Alternativ, wie in diesem Beispiel gezeigt, kann mit der Funktion `then` eine anonyme Funktion z.B. `hls => console.log(hls.toString())` übergeben werden. Diese bekommt den `ColorValue` nach Abschluss der Konvertierung dann übergeben. Anschließend wird der `ColorValue` als `string` in der Konsole ausgegeben.

PixelColorFormatter

Die Funktion `PixelColorFormatter` ist in der Kategorie `Formatting` gelistet. Sie ist der Standardwert für das `PixelColorFormat`-Eigenschaft des Image Watch Controls [▶ 2790]. Sie konvertiert den `PixelColor`-Wert in einen per HTML darstellbaren String. Konkret werden die Farbwerte der einzelnen Kanäle aneinandergehangen und bei bestimmten Farbformaten auch farblich markiert:



Parameter	Datentyp	Beschreibung
`ctx`	`SelectableRequired<TcHmi.Context, 'success' | 'error'> | null`	Wird HMI-intern verwendet, um asynchrone Rückgaben zu empfangen. Kann bei Aufrufen in Javascript aber auf `null` gesetzt werden.
`PixelColor`	`Vector4`	Der zu konvertierende Farbwert.
`Format`	`ColorSpace`	Format für Konvertierung: `Gray, RGB, HSV, HLS, RGBa, HSV FULL, HLS FULL, Lab, YUV, YCrCb`
`Precision`	`Precision`	Legt fest, auf wie viele Dezimalstellen gerundet werden soll. Bei „Unrounded“ wird nicht gerundet.
`PixelType`	`ElementType`	Skalierung der Farbwerte: `8-Bit (USINT), 16-Bit (UINT)`

Die Funktion gibt einen als HTML codierten `String` als `Promise<string>` zurück. Siehe `ConvertColor` zur Handhabung von Promise.

ShapeValueFormatter

Die Funktion `ShapeValueFormatter` ist in der Kategorie `Formatting` gelistet. Sie ist der Standardwert für die `ShapeValueFormat`-Eigenschaft des Image Watch Controls [▶ 2790]. Sie konvertiert den `ShapeValue`-Wert in einen per HTML darstellbaren String. Konkret wird z. B. bei einem Rechteck die Fläche in Pixeln berechnet und als String zurückgegeben:

Parameter	Datentyp	Beschreibung
`ShapeData`	`Any`	Die zu konvertierenden Shape Daten.
`ShapeType`	ShapeType [▶ 2836]	Typ des übergebenen Shapes.

Die Funktion gibt einen als HTML codierten `String` zurück.

ToRotatedRectangle

Die Funktion `ToRotatedRectangle` ist in der Kategorie Data Conversion gelistet. Sie konvertiert ein `UprightRectangle` in ein `RotatedRectangle` mit Winkel `0`.

Parameter	Datentyp	Beschreibung
`rectangle`	TcVnRectangle [▶ 2836]	Das zu konvertierende Rechteck.

Die Funktion gibt das konvertierte Rechteck als TcVnRotatedRectangle [▶ 2836] zurück und kann wie folgt angewandt werden:

```
let upright = {
    nWidth: 100,
    nHeight: 60,
    nX: 200,
    nY: 100,
}
let rotated = TcHmi.Functions.Beckhoff.Vision.ToRotatedRectangle(upright);
console.log(rotated);

//console output
// {
//     aCenter: [250, 130],
//     fAngle: 0,
//     stSize: { fWidth: 100, fHeight: 60 },
// }
```

ToUprightRectangle

Die Funktion `ToUprightRectangle` ist in der Kategorie Data Conversion gelistet. Sie konvertiert ein `RotatedRectangle` in ein `UprightRectangle` und verwirft dabei den Winkel.

Parameter	Datentyp	Beschreibung
`rectangle`	TcVnRotatedRectangle [▶ 2836]	Das zu konvertierende Rechteck.

Die Funktion gibt das konvertierte Rechteck als TcVnRectangle [▶ 2836] zurück und kann wie folgt angewandt werden:

```
let rotated = {
    aCenter: [250, 130],
    fAngle: 0,
    stSize: { fWidth: 100, fHeight: 60 },
}
let upright = TcHmi.Functions.Beckhoff.Vision.ToUprightRectangle(rotated);
console.log(upright);

//console output
// {
//     nWidth: 100,
//     nHeight: 60,
//     nX: 200,
//     nY: 100,
// }
```

7.2.6 Datentypen

ColorChannels

Definiert die Kanäle eines Farbformats. Ist ein Array von Objekten, die folgende Attribute haben und jeweils einen Farbkanal beschreiben:

Eigenschaft	Datentyp	Beschreibung
`color`	`SolidColor`	Anzeigefarbe des Kanals.
`maxValue`	`Number`	Obere Grenze des erlaubten Wertebereichs. Die untere Grenze ist immer 0.
`float`	`Boolean`	Definiert, ob der Kanal durch einen Float oder durch einen Integer beschrieben wird.

Wird z. B. im Color Control [▶ 2822] genutzt.

ColorSymbol

Definiert ein Symbol mit einem ColorValue-kompatiblen Datentyp.

ColorValue

Definiert einen Wert, der als Farbe interpretiert werden kann. Ist entweder eine Zahl, die einen Grauwert beschreibt oder ein 3- oder 4-elementiges Array, welches eine Farbe mit entsprechend vielen Kanälen beschreibt.

Valide Werte sind zum Beispiel:

- `0`
- `128`
- `[0, 0, 255]`
- `[0, 0, 255, 255]`

Orientation

Definiert die Orientierung eines Anzeigeelements. Mögliche Werte sind „Horizontal” und „Vertical“.

TcVnRotatedRectangle

Stellt ein Rechteck dar, welches rotiert sein kann. Ist ein Objekt mit folgenden Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
`aCenter`	`TcVnPoint2_REAL`	Mittelpunkt des Rechtsecks.
`fAngle`	`DINT`	Winkel, um den das Rechteck gedreht ist.
`stSize`	`TcVnSize2_REAL`	Größe des Rechtecks um den Mittelpunkt herum.

ShapeType

Definiert den Typ der auswählbaren Form im Image Watch Control [▶ 2790]. Ist ein Enum mit folgenden Werten:

- Point
- Line
- Square
- Rectangle
- Circle
- Ellipse
- Polygon

TcVnRectangle

Stellt ein Achsen-ausgerichtetes Rechteck dar und ist ein Objekt mit folgenden Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
`nHeight`	`DINT`	Höhe des Rechtecks.
`nWidth`	`DINT`	Breite des Rechtecks.
`nX`	`DINT`	X-Koordinate der oberen rechten Ecke.
`nY`	`DINT`	Y-Koordinate der oberen rechten Ecke.

UsageMode

Definiert die Anzeige und Editierbarkeit von Anzeigeelementen. Mögliche Werte sind:

Wert	Beschreibung
Hidden	Das Element wird nicht angezeigt.
ViewOnly	Das Element wird angezeigt, ist aber nicht editierbar.
Editable	Das Element wird angezeigt und ist editierbar.

Wird hauptsächlich im Color Control [▶ 2817] genutzt.

8 Beispiele und Tools

In diesem Kapitel finden Sie Beispiele zur Anwendung von vielen Funktionen und Funktionsbausteinen in der SPS, sowie von Assistenten und zur Kamerakonfiguration. Bevor Sie sich mit den Beispielen zur SPS beschäftigen, sehen Sie sich zuerst die First Steps [▶ 28] an.

Die Beispiele sind voneinander unabhängig. Jedes Beispiel hat einen Schwerpunkt, unter dem es nachfolgend eingruppiert ist. So sind die Beispiele, die den Schwerpunkt auf jeweils eine Funktion oder einen Funktionsbaustein legen, entsprechend der API-Struktur sortiert.

Weiterhin finden Sie ergänzende Tools zur Unterstützung:

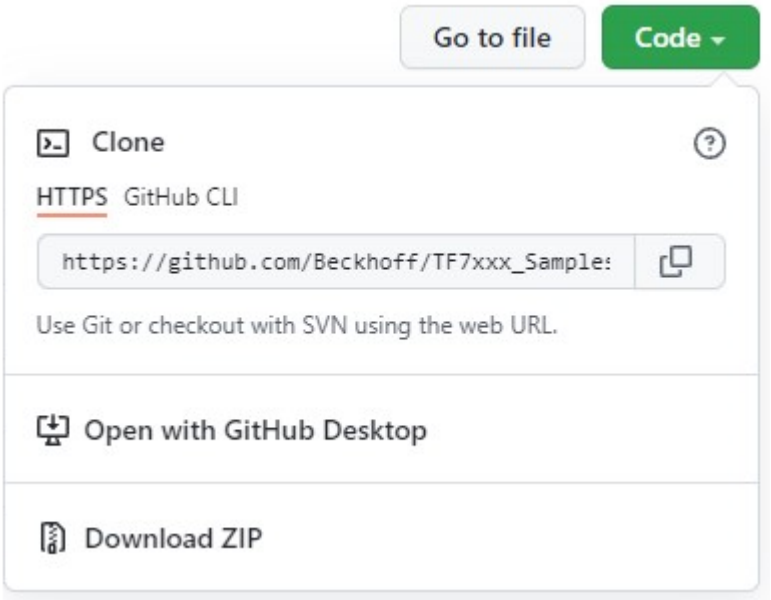
- Neural Network (ONNX) Compatibility Check Tool [▶ 2942]

Download

Die Beispiele können über ein Repository auf GitHub bezogen werden: https://github.com/Beckhoff/TF7xxx_Samples. Sie haben dort die Möglichkeit, das Repository zu clonen oder ein ZIP File mit allen Samples herunterzuladen.

Beispielbilder müssen separat runtergeladen werden

Das GitHub Repository enthält nur die TwinCAT Projekte mit dem entsprechenden Code. Die Bilddaten „TF7xxx_Sample_Images.zip“ müssen separat von der Beckhoff Webseite heruntergeladen werden. Den Link und weitere Informationen finden Sie ebenfalls in der Readme Datei im GitHub Repository.



Projektressourcen

Jedes Beispiel wird in einem separaten Ordner, entsprechend der Gliederungsreihenfolge und mit der Bezeichnung der Kapitelüberschriften, bereitgestellt. Darin befindet sich ein TwinCAT-Projekt mit dem entsprechenden SPS Code.

Den separaten Zip-Ordner mit den Beispielbildern, können Sie z. B. in den Images Order innerhalb des Repository entpacken. Die Ordnernamen sowie die Struktur, in der die Beispielbilder abgelegt sind, ist identisch zu der Projektstruktur. Daher erfolgt die Zuordnung der Bilder einfach über den Pfad mit gleichem Ordnernamen.

Achten Sie auf kurze Pfade zum Beispiel Repository bzw. zu den TwinCAT Projekten

Aufgrund der möglichen Begrenzung der Pfadlänge im Betriebssystem auf 255 Zeichen kann es bei zu langen Ordnerpfaden zu Problemen bei der Ausführung von TwinCAT Projekten kommen. Da die Ordnerstruktur mit den Kapitelüberschriften, zu Gunsten der Verständlichkeit, bereits recht lang ist, sollte ein kurzer Pfad für das komplette Repository oder zu einzelnen TwinCAT Projekten gewählt werden, z.B. C:\TF7xxx_Samples\.

Bevor Sie ein Projekt ausführen, passen Sie die Systemkonfiguration [▶ 32] entsprechend Ihrem Zielsystem an - insbesondere die Konfiguration der Kerne, des Router-Speichers und der SPS-Zykluszeit. Überprüfen Sie weiterhin die Version der TcCOM-Objekte von Kamera oder Filesource sowie der SPS Bibliothek, wie in der Versionsübersicht [▶ 16] beschrieben und passen Sie die Version ggf. an.

Im GitHub Repository finden Sie dazu ein PowerShell Skript (SetTcCOMModuleVersion.ps1) was die Version in allen Projekten des PLC Ordners anpasst. Führen Sie dazu das Skript z. B. über die PowerShell Konsole mit Angabe der gewünschten Version wie folgt aus:

```
PS C:\TF7xxx_Samples> .\SetTcCOMModuleVersion.ps1 4.0.4.8
```

Wenn Bilder oder Streams benötigt werden, binden Sie diese über das File Source Control [▶ 136] oder das Record/Playback [▶ 109] Tab des Kameraobjekts ein.

8.1 Beispiele zu Funktionen

Alle nachfolgenden Beispiele legen ihren Schwerpunkt auf eine einzelne Funktion und sind entsprechend der TwinCAT Vision API-Referenz sortiert. Zu jedem Beispiel gibt es entsprechende Bilder, die über das File Source in die SPS geladen werden. Zusätzlich ist den jeweiligen Beispielbeschreibungen zu entnehmen, welche Funktionsparameter zu ändern sind, um bestimmte Effekte zu sehen.

Die allgemeinen Schritte zum Starten der Funktionsbeispiele sind:

1. Fügen Sie die beigefügten Beispielbilder im File-Source Control hinzu.
   Bei der Verwendung von eigenen Bildern achten Sie auf das erwartete Bildformat, dieses kann je nach Beispiel variieren.
2. Aktivieren Sie die Konfiguration und schalten Sie TwinCAT und die SPS in den Run-Mode.

⇨ Über **View > Other Windows > ADS Image Watch** sehen Sie nun die Bilder aus der SPS. Dabei zeigen
- `ipImageInDisp` immer das Eingangsbild und
- `ipImageResDisp` immer das Ergebnisbild an
(Beide Bildnamen können bei Ihnen natürlich andere sein, wir haben die Variablen in den Beispielen immer gleich benannt, damit sie gut wiedererkannt werden können und Sie sich in jedem Projekt schnell zurecht finden.)
- bei Bedarf zeigen andere Bilder Ergebnisse von Zwischenschritten oder weiterführende Informationen. Details finden Sie in der jeweiligen Beispielbeschreibung.

In den Beispielbeschreibungen finden Sie außerdem die Funktionsparameter, die im eingeloggten Run-

Mode über **Prepared value** und anschließenden Klick auf **write values** zu schreiben sind. Die Effekte sind anschließend wieder in den dargestellten Bildern im ADS Image Watch zu sehen.

Beispiele

- Basic Image Operations [▶ 2842]
- Code Reading [▶ 2845]
- Contour Analysis [▶ 2857]
- Image Analysis [▶ 2867]
- Image Color And Contrast Processing [▶ 2887]
- Image Segmentation [▶ 2897]
- Measurement [▶ 2900]

Beispiel zum Schreiben eigener Funktionen

Wenn Sie für Ihre Applikation bestimmten Abfolgen von TwinCAT Vision API-Funktionen wiederholt nutzen, bietet es sich an, diese in einer Funktion zu kapseln. Folgendes Beispiel erklärt, was Sie dabei beachten sollten:

Beispiel: Selbstgeschriebene Funktionen [▶ 2917]

8.1.1 Basic Container Operations

8.1.1.1 Selektieren von Container-Elementen

In diesem Beispiel werden Container-Elemente mit Hilfe einer benutzerdefinierten Bedingung gefiltert. Dazu wird die Funktion F_VN_CopyContainerElementsConditional_ITcVnContainer [▶ 773] genutzt, da es sich bei den Container-Elementen wiederum um Container handelt. Falls die Container-Elemente Iteratoren sind, kann dazu auch die Funktion F_VN_CopyContainerElementsConditional_ITcVnForwardIterator [▶ 774] genutzt werden.

Erklärung

Wenn die Elemente von mehrdimensionalen Containern gefiltert werden sollen, entsteht durch den Zugriff auf die Sub-Elemente schnell großer Programmieraufwand. Um dem entgegenzuwirken, stellt TwinCAT Vision genau dafür eine Funktionalität bereit, die z.B. wie folgt genutzt wird:

Es wird ein Funktionsblock angelegt, der das Interface `ITcVnCustomElementCondition_ITcVnContainer` implementiert. Dieses Interface besitzt die Methode `Condition`, welche ein Element vom Typ `ITcVnContainer` entgegennimmt. In der Implementierung dieser Methode wird die Filterbedingung festgelegt, die auf das Element angewandt werden soll. Der Funktionsblock mit der Filterbedingung wird dann an die Funktion `F_VN_CopyContainerElementsConditional_ITcVnContainer` übergeben, um die Filterung aller Elemente eines Containers durchzuführen. In diesem Beispiel werden die Elemente danach gefiltert, ob sie mehr als fünf Sub-Elemente besitzen.

Anwendung

Beispielhaft werden die Sub-Container eines Containers danach gefiltert, wie viele Elemente sie jeweils besitzen. Konkret sollen nur die Sub-Container in einen neuen Container übernommen werden, die mehr als fünf Elemente haben. Die Struktur des Containers kann mittels des Arrays `aContainerStructure` variiert werden. Die Elemente geben an, wie viele Sub-Elemente jeder Sub-Container hat. Wenn ebenfalls die Anzahl der der Sub-Container selbst variiert wird, muss die Konstante `cNumberOfSubContainers` darauf abgestimmt werden.

Programm

MAIN

Zunächst wird ein konstanter Integer definiert, um die Anzahl der Sub-Container einfach variieren zu können:

```
VAR CONSTANT
    cNumberOfSubContainers  :   INT := 5;
END_VAR
```

Daraufhin werden alle benötigten Variablen deklariert:

```
VAR
    aContainerStructure     :   ARRAY [0..(cNumberOfSubContainers-1)] OF INT := [2, 7, 3, 5, 12];
    ipHelper                :   ITcVnContainer;
    ipContainerBase         :   ITcVnContainer;
    ipContainerFiltered     :   ITcVnContainer;
    fbCondition             :   FB_ConditionMoreThanFive;
    nSelectedContainers     :   ULINT;
    hr                      :   HRESULT;
    i                       :   INT;
END_VAR
```

aContainerStructure	Definition der Container-Struktur. Ein Array der Länge `5` mit den Initialisierungswerten `[2, 7, 3, 5, 12]` bedeutet z. B., dass im Beispiel ein Container mit 5 Sub-Containern erstellt werden soll. Dabei soll der erste Sub-Container 2 Elemente haben, der zweite Sub-Container 7 Elemente haben, usw. Die Länge des Arrays wird durch die Konstante `cNumberOfSubContainers` definiert.
ipHelper	Hilfs-Interface-Pointer, der dazu benutzt wird, den Hauptcontainer `ipContainer` mit Sub-Containern zu füllen.
ipContainerBase	Container, in dem die in `aNumberOfElement` definierten Sub-Container erstellt werden.
fbCondition	Funktionsblock, der die Filterbedingung enthält.
ipContainerFiltered	Container, der am Ende nur die Sub-Container aus `ipContainer` enthalten soll, die die Filterbedingung erfüllen.
nSelectedContainers	Integer, der anzeigt, wie viele Sub-Container die Filterbedingung tatsächlich erfüllen.
hr	Statusvariable vom Typ HRESULT [▶ 150].
i	Hilfsvariable

Im Programmteil wird ein Container erstellt, der mit den oben definierten Sub-Containern gefüllt wird. Der Aufruf von F_VN_ReserveContainerMemory [▶ 793] ist dabei optional und dient nur einer besseren Performance.

```
hr := F_VN_CreateContainer(ipContainerBase, ContainerType_Vector_Vector_REAL, 0, hr);
hr := F_VN_ReserveContainerMemory(ipContainerBase, cNumberOfSubContainers, hr);
FOR i:=0 TO (cNumberOfSubContainers-1) DO
    hr := F_VN_CreateContainer(ipHelper, ContainerType_Vector_REAL, aContainerStructure[i], hr);
    hr := F_VN_AppendToContainer_ITcVnContainer(ipHelper, ipContainerBase, hr);
END_FOR
```

Daraufhin wird der gerade erstellte Container gefiltert. Als Filterbedingung wird der Funktionsblock mit der selbst-geschriebenen Bedingung angegeben.

```
hr := F_VN_CopyContainerElementsConditional_ITcVnContainer(
    ipSrcContainer      :=  ipContainerBase,
    ipDestContainer     :=  ipContainerSelection,
    ipConditionFB       :=  fbCondition,
    hr);
```

Um die Funktionalität der Funktion nachzuvollziehen, werden die Elemente, die die geschriebene Bedingung erfüllen, gezählt. Die Variable `nSelectedContainers` ändert sich nun abhängig von den Sub-Containern und der Filterbedingung.

```
hr := F_VN_GetNumberOfElements(ipContainerSelection, nSelectedContainers, hr);
```

In diesem Fall gibt die Anzahl an, wie viele der Sub-Container mehr als fünf Sub-Elemente haben.

Funktionsblock mit Filterbedingung

Die Filterbedingung wird als Funktionsblock mit einer Methode `Condition` übergeben. Damit ein selbst-geschriebener Funktionsblock an die Funktion übergeben werden kann, muss dieser das Interface ITcVnCustomElementCondition_ITcVnContainer [▶ 261] implementieren:

```
FUNCTION_BLOCK FB_ConditionMoreThanFive IMPLEMENTS ITcVnCustomElementCondition_ITcVnContainer
```

Die Methode `Condition` definiert die Filterbedingung. Sie wird für jeden Sub-Container einzeln aufgerufen. Je nachdem, ob der Rückgabewert `TRUE` oder `FALSE` ist, wird der jeweilige Sub-Container in den neuen Container übernommen oder nicht.

```
METHOD Condition : BOOL
```

Bei jedem Aufruf der Methode wird die aktuelle Sub-Container als Eingangsparameter übergeben:

```
VAR_INPUT
    ipElement           :   ITcVnContainer;
END_VAR
```

In diesem Fall soll ein Element übernommen werden, wenn es mehr als fünf Sub-Elemente hat. Dafür wird die Anzahl der Sub-Elemente berechnet und abhängig vom Ergebnis der Wert `TRUE` oder `FALSE` zurückgegeben.

```
hr := F_VN_GetNumberOfElements(ipElement, nNumberOfElements, hr);
IF FAILED(hr) OR nNumberOfElements <= 5 THEN
    Condition := FALSE;
ELSE
    Condition := TRUE;
END_IF
```

Die benötigten Variablen sind wie folgt deklariert:

```
VAR
    nNumberOfElements   :   ULINT;
    hr                  :   HRESULT;
END_VAR
```

Sowohl der Funktionsblock selbst als auch die Methode `Condition` müssen das Compiler-Attribut `{attribute 'c++_compatible'}` enthalten.

8.1.2 Basic Image Operations

8.1.2.1 Mehrere Bilder zusammenfügen

In diesem Beispiel werden fünf Bilder mit Hilfe der Funktion F_VN_FuseImagesArray [▶ 817] aneinandergehängt. Diese Funktionalität ist vor allem in Verbindung mit Zeilenkameras und der nahtlosen Aufnahme von kontinuierlichem Materialfluss hilfreich.

Erklärung

Die Funktion F_VN_FuseImagesArray [▶ 817] nimmt ein Array von (maximal zehn) Bildern entgegen und fügt diese in einem neuen Bild aneinander. In dem folgenden Beispiel-Programm werden zunächst fünf unterschiedlich gefärbte Bilder mit einer Höhe von 10px erstellt. Die Anzahl kann durch die Variable *nImages* und die Höhe mit *nHeight* verändert werden. Achten Sie darauf, die Anzahl nur im Bereich 1-10 zu variieren.

Die Größe des Ergebnisbildes wird durch Angabe der Bildzeilen mit den Funktionsparametern *nFirstLine* und *nNumLines* festgelegt. Dadurch ist es möglich, das erste und letzte Bild „anzuschneiden“.

Falls nur zwei Bilder zusammengefügt werden sollen, kann ebenfalls die Funktion F_VN_FuseImages [▶ 815] verwendet werden.

Programm

Folgende Variablen werden deklariert:

```
VAR
    hr                  :  HRESULT;
    aImages             :  ARRAY[0..9] OF ITcVnImage;
    ipImageFused        :  ITcVnImage;
    ipImageFusedDisp    :  ITcVnDisplayableImage;
    nImages             :  INT := 5;
    nHeight             :  UDINT := 10;
    aColor              :  ARRAY[0..9] OF TcVnVector4_LREAL := [[255,0,0], [200,50,0], [150,100,0],
[100,150,0], [50,200,0], [0,250,0], [0,200,50], [0,150,100], [0,100,150], [0,50,200]];
    i                   :  INT;
VAR
```

hr	Statusvariable vom Typ HRESULT [▶ 150].
aImages	Array mit mehreren Bildern, die zusammengefügt werden sollen.
ipImageFused	Ergebnisbild, in dem mehrere Bilder aus aImages zusammengefügt werden.
ipImageFusedDisp	Anzeigbares Ergebnisbild.
nImages	Anzahl der Bilder, die zusammengefügt werden sollen.
aColor	Zehn verschiedene Farben, um die Bilder in aImages visuell auseinanderhalten zu können.
i	Hilfsvariable.

Im Programm werden zunächst die Bilder in dem Array `aImages` initialisiert und eingefärbt. Um die Bilder visuell auseinanderhalten zu können, werden zehn verschiedene Farben aus `aColor` verwendet.

```
FOR i := 0 TO nImages-1 DO
     hr := F_VN_CreateImageAndSetPixels(aImages[i], 100, nHeight, ETcVnElementType.TCVN_ET_USINT, 3,
aColor[i], hr);
END_FOR
```

Daraufhin wird eine definierte Anzahl `nImages` an Bildern aus `aImages` in dem Bild `ipImageFused` vertikal zusammengefügt:

```
hr := F_VN_FuseImagesArray(aImages, nImages, ipImageFused, 0, nImages * nHeight, hr);
```

Schließlich wird das Bild angezeigt:

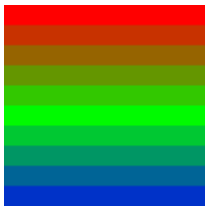
```
hr := F_VN_TransformIntoDisplayableImage(ipImageFused, ipImageFusedDisp, hr);
```

Ergebnis

nImages = 2:

nImages = 5:

nImages = 10:



Ähnliche Beispiele

Bildbereiche kopieren [▶ 2843]

8.1.2.2 Bildbereiche kopieren

In diesem Beispiel werden zwei Bilder in ein anderes Bild kopiert, wobei verschiedene Methoden genutzt werden:

- Nutzung der Funktion F_VN_CopyImageRegionToRegion [▶ 802],
- Auswahl einer ROI und Kopieren eines Bildes und
- Auswahl einer ROI und Addieren von zwei Bildern mittels der Funktion F_VN_AddImages [▶ 524].

Erläuterung

Mit den unten gezeigten Methoden können beliebige, rechteckige Bildbereiche zwischen Bildern ausgetauscht werden. Bei Bedarf kann dies mit Hilfe von arithmetischen Bild-Operationen auch teil-transparent geschehen.

Die Funktion F_VN_CopyImageRegionToRegion [▶ 802] nimmt zwei Bilder und entsprechend zwei Bildregionen entgegen und kopiert den Inhalt der einen Bildregion entsprechend in die andere. Beide Bildregionen haben die gleiche Größe und müssen jeweils vollständig im Bild vorhanden sein. Die Funktion wird im Beispiel zum Kopieren des roten Bildes in das Mischbild verwendet. Zum Kopieren des blauen Bildes wird hingegen eine ausprogrammierte Version mit gleicher Funktionalität genutzt. Der Vorteil dieser Version ist, dass anstatt einer Kopie auch eine Vermischung des blauen und des Mischbilds möglich ist.

Im Beispiel kann die Position der beiden kopierten Bilder über die Variablen `aPositionRed` und `aPositionBlue` verändert werden. Der Parameter `bCopyBlue` bestimmt, ob das blaue Bild kopiert oder vermischt wird.

Variablen

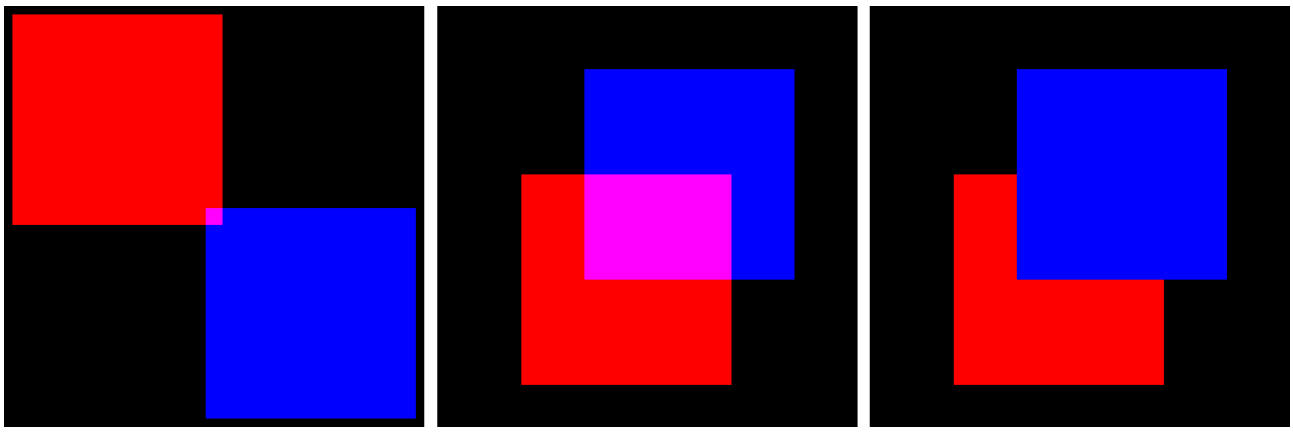
```
hr                :   HRESULT;

ipImageMerge      :   ITcVnImage;
ipImageRed        :   ITcVnImage;
ipImageBlue       :   ITcVnImage;
ipImageMergeDisp  :   ITcVnDisplayableImage;

aBlack            :   TcVnVector4_LREAL := [0, 0, 0];
aRed              :   TcVnVector4_LREAL := [255, 0, 0];
aBlue             :   TcVnVector4_LREAL := [0, 0, 255];

aPositionRed      :   TcVnPoint := [20, 20];
aPositionBlue     :   TcVnPoint := [480, 480];
bCopyBlue         :   BOOL := FALSE;
```

Programm

```
// create images & set colors
hr := F_VN_CreateImageAndSetPixels(ipImageMerge, 1000, 1000, TCVN_ET_USINT, 3, aBlack, hr);
hr := F_VN_CreateImageAndSetPixels(ipImageRed, 500, 500, TCVN_ET_USINT, 3, aRed, hr);
hr := F_VN_CreateImageAndSetPixels(ipImageBlue, 500, 500, TCVN_ET_USINT, 3, aBlue, hr);

// Copy red image to selected region in merge-image
hr := F_VN_CopyImageRegionToRegion(
    ipSrcImage  :=  ipImageRed,
    nXSrc       :=  0,      nYSrc   :=  0,
    nWidth      :=  500,    nHeight :=  500,
    ipDestImage :=  ipImageMerge,
    nXDest      :=  aPositionRed[0],
    nYDest      :=  aPositionRed[1],
    hrPrev      :=  hr
);

// Copy or mix blue image to selected region in merge-image
hr := F_VN_SetRoi(aPositionBlue[0], aPositionBlue[1], 500, 500, ipImageMerge, hr);
IF bCopyBlue THEN
    hr := F_VN_CopyImage(ipImageBlue, ipImageMerge, hr);
ELSE
    hr := F_VN_AddImages(ipImageBlue, ipImageMerge, ipImageMerge, hr);
END_IF
hr := F_VN_ResetRoi(ipImageMerge, hr);

hr := F_VN_TransformIntoDisplayableImage(ipImageMerge, ipImageMergeDisp, hr);
```

Ergebnis

Folgend sind beispielhaft Ergebnisse des Programms abgebildet. Bei den Bildern Links und Mitte wurde das blaue Quadrat addiert, beim Bild Rechts wurde es kopiert. Zudem sind die Ziel-Positionen der Quadrate unterschiedlich.



Ähnliche Beispiele

Mehrere Bilder zusammenfügen [▶ 2842]

8.1.3 Code Reading

8.1.3.1 EAN-13 Barcode Reading

In diesem Beispiel werden

- über die Funktion F_VN_ReadBarcodeExp [▶ 867] jeweils ein EAN13-Code pro Bild ausgelesen
- und die Ausführungszeit der Funktion über einen sogenannten Watchdog [▶ 156] überwacht.

Erläuterung

- Die Funktionen F_VN_ReadBarcode [▶ 865] / F_VN_ReadBarcodeExp [▶ 867] können Barcodes verschiedener Typen im Bild suchen und auslesen. Zur Beschleunigung der Verarbeitungszeit kann der Barcode-Typ (`eBarcodeType`) vorgegeben werden. In diesem Beispiel ist das `TCVN_BT_EAN13`. Zusätzlich kann die Verarbeitungszeit über den sogenannten Watchdog überwacht werden. Zum Watchdog gibt es das vertiefende Beispiel Blob Detection mit Watchdog Überwachung [▶ 2867].
- Die Funktion F_VN_ReadBarcodeExp [▶ 867], die in diesem Beispiel verwendet wird, bietet im Vergleich zur F_VN_ReadBarcode [▶ 865] zudem die Möglichkeit:
  - zusätzlich zum Barcode-Typ auch die Suchrichtung (`eSearchDirection`) vorzugeben:
    - bei TCVN_BSD_ANY wird zunächst in horizontaler Richtung und anschließend in vertikaler Richtung gesucht
    - bei TCVN_BSD_HORIZONTAL wird nur in horizontaler Richtung gesucht
    - bei TCVN_BSD_VERTICAL wird nur in vertikaler Richtung gesucht
  - den Codebereich zurückgegeben, in dem das Auslesen erfolgte.

Variablen

```
hr                          :   HRESULT;

ipImageIn                   :   ITcVnImage;
ipImageInDsply              :   ITcVnDisplayableImage;

ipImageRes                  :   ITcVnImage;
ipImageResDsply             :   ITcVnDisplayableImage;

// Barcode
ipCodeDecodedList           :   ITcVnContainer;
ipCodeContourList           :   ITcVnContainer;
sCodeAsString               :   STRING(255);
eBarcodeSearchDirection     :   ETcVnBarcodeSearchDirection := TCVN_BSD_ANY;
eBarcodeType                :   ETcVnBarcodeType := TCVN_BT_EAN13;

// Watchdog
hrWD                        :   HRESULT;
tStop                       :   DINT := 50000;
```

```
tRest                       :   DINT;

// Output
sText                       :   STRING;

// Color
aColorRed                   :   TcVnVector4_LREAL := [255, 0, 0];
```

Code

```
// Execute the Barcode Reading Function with EAN13 selected monitored by the Watchdog-Function
// ---------------------------------------------------------------
hrWD := F_VN_StartRelWatchdog(tStop, S_OK);
    hr := F_VN_ReadBarcodeExp(
                ipSrcImage          :=   ipImageIn,
                ipDecodedData       :=   ipCodeDecodedList,
                ipContours          :=   ipCodeContourList,
                eBarcodeType        :=   eBarcodeType,
                nCodeNumber         :=   1,
                eSearchDirection    :=   eBarcodeSearchDirection,
                hrPrev              :=   hr);
hrWD := F_VN_StopWatchdog(hrWD, tRest => tRest);

// Check if the function was executed successfully
IF hr = S_OK THEN
    // Export Code into String
    hr := F_VN_ExportSubContainer_String(ipCodeDecodedList, 0, sCodeAsString, 255, hr);

    // Write Code into Result Image
    hr := F_VN_PutTextExp(sCodeAsString, ipImageRes, 50, 100, ETcVnFontType.TCVN_FT_HERSHEY_PLAIN,
5, aColorRed,3, TCVN_LT_4_CONNECTED, FALSE, hr);

    // Draw Code Contour into Result Image
    hr := F_VN_DrawContours(ipCodeContourList, 0, ipImageRes, aColorRed, 3, hr);
ELSE
    // Write HRESULT into Result Image
    sText := CONCAT('Returncode ', DINT_TO_STRING(hr));
    hr := F_VN_PutTextExp(sText, ipImageRes, 50, 100, ETcVnFontType.TCVN_FT_HERSHEY_PLAIN, 5,
aColorRed,3, TCVN_LT_4_CONNECTED, FALSE, hr);
END_IF

// Write Code Reading proceeded time into Result Image
sText := CONCAT(CONCAT('Time: ', DINT_TO_STRING(tStop - tRest)), 'us');
hr := F_VN_PutTextExp(sText, ipImageRes, 50, 200, ETcVnFontType.TCVN_FT_HERSHEY_PLAIN, 5,
aColorRed,3, TCVN_LT_4_CONNECTED, FALSE, hr);
```

Ergebnisse

Wenn als Suchrichtung TCVN_BSD_ANY ausgewählt ist, werden sowohl horizontale als auch vertikale EAN13-Codes erkannt. Beim Vergleich der Watchdog-Zeit wird aber ersichtlich, dass horizontale EAN13-Codes (erstes Bild) deutlich schneller erkannt werden als vertikale EAN13-Codes (zweites Bild), da zunächst in horizontaler Richtung nach Codes gesucht wird.

Wird hingegen die Suchrichtung auf TCVN_BSD_VERTICAL gesetzt, ist der vertikale EAN13 Code deutlich schneller erkannt (drittes Bild). Es erfolgt dann aber keine Suche nach den horizontalen Codes.

Ähnliche Beispiele

- Data Matrix Code Reading [▶ 2848]
- Pharma-Code Reading [▶ 2850]
- QR-Code Reading [▶ 2852]
- Ergebnisauswertung beim Code Reading [▶ 2854]

8.1.3.2 Data Matrix Code Reading

In diesem Beispiel werden

- über die Funktion F_VN_ReadDataMatrixCodeExp [▶ 874] jeweils ein Data Matrix Code pro Bild ausgelesen
- und die Ausführungszeit der Funktion über einen sogenannten Watchdog [▶ 156] überwacht.

Erläuterung

Die Funktion F_VN_ReadDataMatrixCodeExp [▶ 874] sucht nach Data Matrix Codes im vorgegebenen Bild und liest diese aus. Dabei kann im Gegensatz zur Funktion F_VN_ReadDataMatrixCode [▶ 871] die Anzahl der zu suchenden Codes je Bild vorgegeben werden. In diesem Beispiel ist das ein Code pro Bild. Zudem wird die Kontur des gefundenen Codes zurückgegeben. Als Suchstrategie wird `TCVN_CSS_ONLY_NOT_FLIPPED` gewählt, da das Bild nicht gespiegelt ist.

Wenn sich im Bild Bereiche befinden, die ähnlich wie Codes aussehen, oder wenn gar kein Code im Bild vorhanden ist, kann sich dies negativ auf die Ausführungszeit auswirken. Um Zykluszeitüberschreitungen zu verhindern, wird die Funktion F_VN_ReadDataMatrixCodeExp [▶ 874] in diesem Beispiel mit Hilfe eines Watchdogs [▶ 156] überwacht (siehe dazu das Beispiel Blob Detection mit Watchdog Überwachung [▶ 2867]).

Variablen

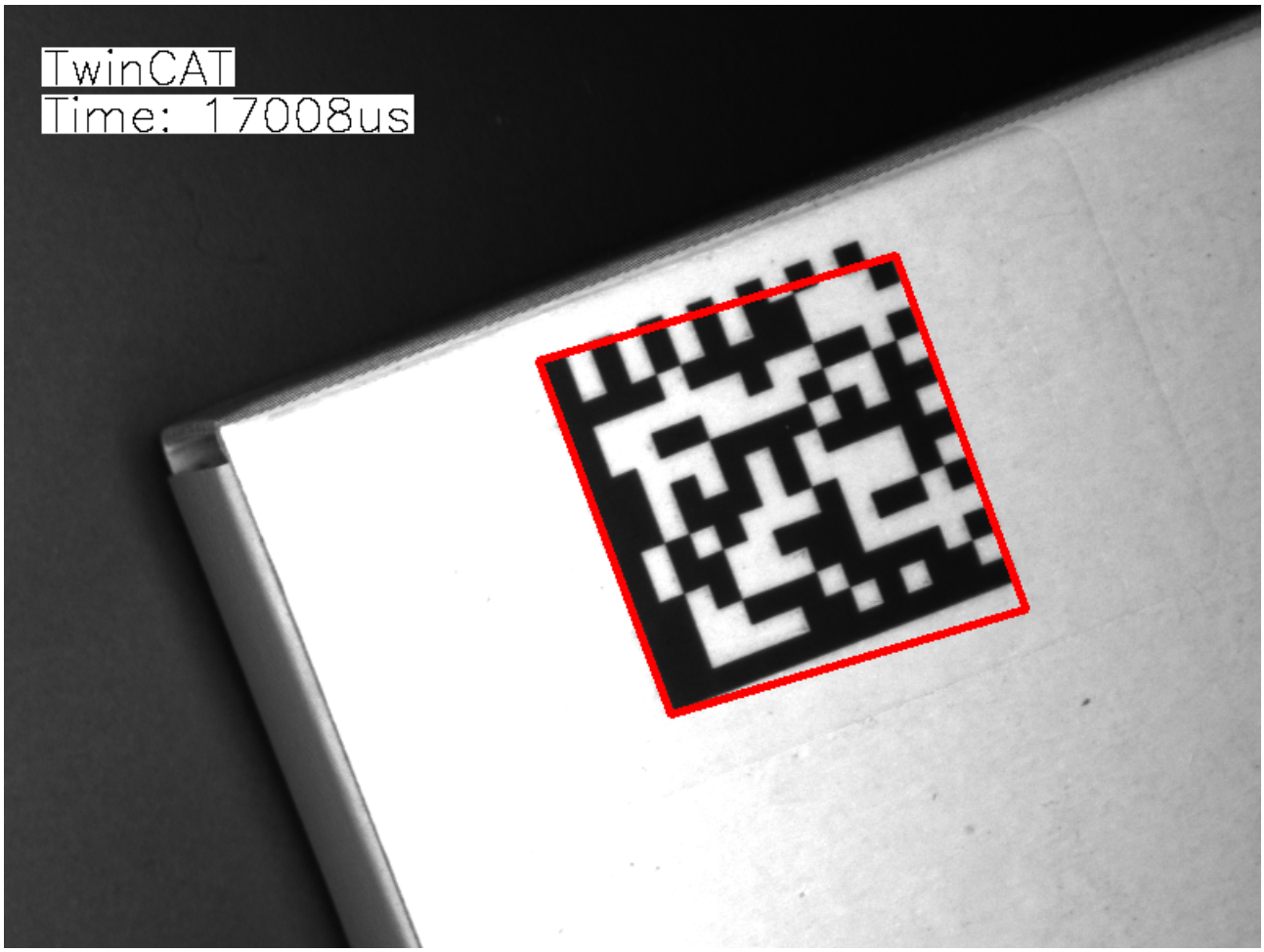
```
hr                  :    HRESULT;

ipImageIn           :    ITcVnImage;
ipImageInDisp       :    ITcVnDisplayableImage;
```

```
ipImageRes          :    ITcVnImage;
ipImageResDisp      :    ITcVnDisplayableImage;

// Data Matrix Code
ipCodeDecodedList   :    ITcVnContainer;
ipCodeContourList   :    ITcVnContainer;
sCodeAsString       :    STRING(255);

// Watchdog
hrWD                :    HRESULT;
tStop               :    DINT := 20000;
tRest               :    DINT;

// Output
sText               :    STRING;

// Color
aColorRed           :    TcVnVector4_LREAL := [255, 0, 0];
```

Code

```
// Execute the DMC Code Reading Function monitored by the Watchdog function
// ---------------------------------------------------------------
hrWD := F_VN_StartRelWatchdog(tStop, S_OK);
    hr := F_VN_ReadDataMatrixCodeExp(
        ipSrcImage      :=  ipImageIn,
        ipDecodedData   :=  ipCodeDecodedList,
        ipContours      :=  ipCodeContourList,
        nCodeNumber     :=  1,
        eSearchStrategy :=  TCVN_CSS_ONLY_NOT_FLIPPED,
        hrPrev          :=  hr
    );
hrWD := F_VN_StopWatchdog(hrWD, tRest => tRest);

// Check if the function was executed successfully
IF hr = S_OK THEN
    // Export Code into String
    hr := F_VN_ExportSubContainer_String(ipCodeDecodedList, 0, sCodeAsString, 255, hr);

    // Write Code into Result Image
    hr := F_VN_PutText(sCodeAsString, ipImageRes, 25, 50, ETcVnFontType.TCVN_FT_HERSHEY_PLAIN, 2,
aColorRed, hr);

    // Draw Code Contour into Result Image
    hr := F_VN_DrawContours(ipCodeContourList, 0, ipImageRes, aColorRed, 3, hr);
ELSE
    // Write HRESULT into Result Image
    sText := CONCAT('Returncode ', DINT_TO_STRING(hr));
    hr := F_VN_PutLabelExp(sText, ipImageRes, 25, 50, 2, 2, ETcVnFontType.TCVN_FT_HERSHEY_PLAIN,
aColorRed, aColorWhite, ETcVnLineType.TCVN_LT_8_CONNECTED, hr);
END_IF

// Write Code Reading proceeded time into Result Image
sText := CONCAT(CONCAT('Time: ', DINT_TO_STRING(tStop - tRest)), 'us');
hr := F_VN_PutText(sText, ipImageRes, 25, 80, ETcVnFontType.TCVN_FT_HERSHEY_PLAIN, 2, aColorRed,
hr);
```

Ergebnis



Ähnliche Beispiele

- EAN-13 Barcode Reading [▶ 2845]
- Pharma-Code Reading [▶ 2850]
- QR-Code Reading [▶ 2852]
- Ergebnisauswertung beim Code Reading [▶ 2854]

8.1.3.3 Pharma-Code Reading

In diesem Beispiel werden

- über die Funktion F_VN_ReadPharmaCodeExp [▶ 882] jeweils ein Pharma-Code pro Bild ausgelesen
- und die Ausführungszeit der Funktion über einen Watchdog [▶ 156] überwacht.

Erläuterung

- Die Funktion F_VN_ReadPharmaCodeExp [▶ 882] sucht nach Pharma-Codes im vorgegebenen Bild und liest diese aus. Dabei wird im Gegensatz zur Funktion F_VN_ReadPharmaCode [▶ 880] zusätzlich die Codeposition zurückgegeben.
- Codeähnliche Strukturen wirken sich negativ auf die Ausführungszeit aus. Um Zykluszeitüberschreitungen zu verhindern, wird die Funktion F_VN_ReadPharmaCodeExp [▶ 882] im Beispiel über den Watchdog [▶ 156] überwacht. Das Beispiel Blob Detection mit Watchdog Überwachung [▶ 2867] erklärt die Benutzung eines Watchdogs [▶ 156].

Variablen

```
hr                  :    HRESULT;

ipImageIn           :    ITcVnImage;
ipImageInDisp       :    ITcVnDisplayableImage;

ipImageRes          :    ITcVnImage;
ipImageResDisp      :    ITcVnDisplayableImage;

// Pharma-Code
ipCodeDecodedList   :    ITcVnContainer;
ipCodeContourList   :    ITcVnContainer;
sCodeAsString       :    STRING(255);
aCodeContour        :    ARRAY[0..3] OF TcVnPoint2_DINT;

// Watchdog
hrWD                :    HRESULT;
tStop               :    DINT := 5000;
tRest               :    DINT;

// Output
sText               :    STRING;

// Color
aColorRed           :    TcVnVector4_LREAL := [255, 0, 0];
```

Code

```
// Execute the Pharma-Code Reading Function monitored by the Watchdog-Function
// ---------------------------------------------------------------
hrWD := F_VN_StartRelWatchdog(tStop, S_OK);
    hr := F_VN_ReadPharmaCodeExp(
                        ipSrcImage      := ipImageIn,
                        ipDecodedData   := ipCodeDecodedList,
                        ipContours      := ipCodeContourList,
                        nCodeNumber     := 1,
                        nMinBarNumber   := 4,
                        hrPrev          := hr);
hrWD := F_VN_StopWatchdog(hrWD, tRest => tRest);

// Check if the function was executed successfully
IF hr = S_OK THEN
    // Export Code into String
    hr := F_VN_ExportSubContainer_String(ipCodeDecodedList, 0, sCodeAsString, 255, hr);

    // Write Code into Result Image
    sText := CONCAT('Code: ', sCodeAsString);
    hr := F_VN_PutTextExp(sText, ipImageRes, 25, 50, ETcVnFontType.TCVN_FT_HERSHEY_PLAIN, 3,
aColorRed, 2, TCVN_LT_4_CONNECTED, FALSE, hr);

    // Draw Code Contour into Result Image
    hr := F_VN_DrawContours(ipCodeContourList, 0, ipImageRes, aColorRed, 3, hr);
ELSE
    // Write HRESULT into Result Image
    sText := CONCAT('Returncode ', DINT_TO_STRING(hr));
    hr := F_VN_PutTextExp(sText, ipImageRes, 25, 50, ETcVnFontType.TCVN_FT_HERSHEY_PLAIN, 3,
aColorRed,2, TCVN_LT_4_CONNECTED, FALSE, hr);
END_IF

// Write Code Reading proceeded time into Result Image
sText := CONCAT(CONCAT('Time: ', DINT_TO_STRING(tStop - tRest)), 'us');
hr := F_VN_PutTextExp(sText, ipImageRes, 25, 100, ETcVnFontType.TCVN_FT_HERSHEY_PLAIN, 3,
aColorRed,2, TCVN_LT_4_CONNECTED, FALSE, hr);
```

Ergebnis

Ähnliche Beispiele

- Data Matrix Code Reading [▶ 2848]
- EAN-13 Barcode Reading [▶ 2845]
- QR-Code Reading [▶ 2852]
- Ergebnisauswertung beim Code Reading [▶ 2854]

8.1.3.4 QR-Code Reading

In diesem Beispiel werden

- über die Funktion F_VN_ReadQRCodeExp [▶ 888] jeweils ein QR-Code pro Bild ausgelesen
- und die Ausführungszeit der Funktion über einen sogenannten Watchdog [▶ 156] überwacht.

Erläuterung

Die Funktion F_VN_ReadQRCodeExp [▶ 888] sucht nach QR-Codes im vorgegebenen Bild und liest diese aus. Dabei wird im Gegensatz zur Funktion F_VN_ReadQRCode [▶ 885] zusätzlich die Kontur des gefundenen Codes zurückgegeben.

Wenn sich im Bild Bereiche befinden, die ähnlich wie Codes aussehen, oder wenn gar kein Code im Bild vorhanden ist, kann sich dies negativ auf die Ausführungszeit auswirken. Um Zykluszeitüberschreitungen zu verhindern, wird die Funktion F_VN_ReadQRCodeExp [▶ 888] im Beispiel über mit Hilfe eines Watchdogs [▶ 156] überwacht (siehe dazu das Beispiel Blob Detection mit Watchdog Überwachung [▶ 2867]).

Variablen

```
hr                  :    HRESULT;

ipImageIn           :    ITcVnImage;
ipImageInDisp       :    ITcVnDisplayableImage;

ipImageRes          :    ITcVnImage;
ipImageResDisp      :    ITcVnDisplayableImage;

// QR Code
ipCodeDecodedList   :    ITcVnContainer;
ipCodeContourList   :    ITcVnContainer;
sCodeAsString       :    STRING(255);

// Watchdog
hrWD                :    HRESULT;
tStop               :    DINT := 20000;
tRest               :    DINT;

// Output
sText               :    STRING;

// Color
aColorRed           :    TcVnVector4_LREAL := [255, 0, 0];
```

Code

```
// Execute the QR Code Reading Function monitored by the Watchdog-Function
// ---------------------------------------------------------------
hrWD := F_VN_StartRelWatchdog(tStop, S_OK);
    hr := F_VN_ReadQRCodeExp(
            ipSrcImage      :=  ipImageIn,
            ipDecodedData   :=  ipCodeDecodedList,
            ipContours      :=  ipCodeContourList,
            nCodeNumber     :=  1,
            eSearchStrategy :=  TCVN_CSS_ONLY_NOT_INVERTED + TCVN_CSS_ONLY_NOT_FLIPPED,
            hrPrev          :=  hr
    );
hrWD := F_VN_StopWatchdog(hrWD, tRest => tRest);

// Check if the function was executed successfully
IF hr = S_OK THEN
    // Export Code into String
    hr := F_VN_ExportSubContainer_String(ipCodeDecodedList, 0, sCodeAsString, 255, hr);

    // Write Code into Result Image
    hr := F_VN_PutTextExp(sCodeAsString, ipImageRes, 50, 100, ETcVnFontType.TCVN_FT_HERSHEY_PLAIN,
5, aColorRed,3, TCVN_LT_4_CONNECTED, FALSE, hr);

    // Draw Code Contour into Result Image
    hr := F_VN_DrawContours(ipCodeContourList, 0, ipImageRes, aColorRed, 3, hr);
ELSE
    // Write HRESULT into Result Image
    sText := CONCAT('Returncode ', DINT_TO_STRING(hr));
    hr := F_VN_PutTextExp(sText, ipImageRes, 50, 100, ETcVnFontType.TCVN_FT_HERSHEY_PLAIN, 5,
aColorRed,3, TCVN_LT_4_CONNECTED, FALSE, hr);
END_IF

// Write Code Reading proceeded time into Result Image
sText := CONCAT(CONCAT('Time: ', DINT_TO_STRING(tStop - tRest)), 'us');
hr := F_VN_PutTextExp(sText, ipImageRes, 50, 200, ETcVnFontType.TCVN_FT_HERSHEY_PLAIN, 5,
aColorRed,3, TCVN_LT_4_CONNECTED, FALSE, hr);
```

Ergebnis

Ähnliche Beispiele

- Data Matrix Code Reading [▶ 2848]
- EAN-13 Barcode Reading [▶ 2845]
- Pharma-Code Reading [▶ 2850]
- Ergebnisauswertung beim Code Reading [▶ 2854]

8.1.3.5 Ergebnisauswertung beim Code Reading

In diesem Beispiel werden

- über die Funktion F_VN_ReadBarcode [▶ 867] jeweils ein EAN13-Code pro Bild ausgelesen,
- die Ausführungszeit der Funktion über einen sogenannten Watchdog [▶ 156] überwacht
- und unterschiedliche Return-Codes der Funktion behandelt.

Erläuterung

- Die Funktion F_VN_ReadBarcode [▶ 865] kann Barcodes verschiedener Typen im Bild suchen und auslesen. Details zur Verwendung der Funktion finden Sie im Beispiel EAN-13 Barcode Reading [▶ 2845].
- Die Behandlung des Return-Codes der Funktion wird anhand einer entsprechenden Abfrage erläutert. Dabei wird zwischen den folgenden Fällen unterschieden:
  - Erfolgreiche Ausführung der Funktion, Code wurde gefunden und dekodiert.
  - Erfolgreiche Ausführung der Funktion, Code ist nicht im Bild vorhanden oder kann nicht gefunden / dekodiert werden.
  - Die Funktion hat selbst keinen Fehler und wurde vom Watchdog abgebrochen.
  - Funktion hat sonstigen Fehler.

Eine genaue Erläuterung der Return-Codes für Code Reading Funktionen finden Sie im Übersichtskapitel Code Reading [▶ 863].

Variablen

```
hr                          :   HRESULT;

// Images
ipImageIn                   :   ITcVnImage;
ipImageInDisp               :   ITcVnDisplayableImage;
ipImageRes                  :   ITcVnImage;
ipImageResDisp              :   ITcVnDisplayableImage;

// Barcode
ipCodeDecodedList           :   ITcVnContainer;
ipCodeContourList           :   ITcVnContainer;
sCodeAsString               :   STRING(255);

// Watchdog
hrWD                        :   HRESULT;
tStop                       :   DINT := 50000;
tRest                       :   DINT;

// Color
aColorGreen                 :   TcVnVector4_LREAL := [0, 255, 0];

// Return code
nReturnCode                 :   DWORD;
sReturnCode                 :   STRING;
sResultText                 :   STRING;
```

Code

```
// Execute the Barcode Reading Function with EAN13 selected monitored by the Watchdog-Function
hrWD := F_VN_StartRelWatchdog(tStop, S_OK);
    hr := F_VN_ReadBarcode(
        ipSrcImage      :=  ipImageIn,
        ipDecodedData   :=  ipCodeDecodedList,
        eBarcodeType    :=  ETcVnBarcodeType.TCVN_BT_EAN13,
        hrPrev          :=  hr
    );
hrWD := F_VN_StopWatchdog(hrWD, tRest => tRest);

// Handle return-code
IF hr = S_OK THEN
    // Prepare code result in string
    hr := F_VN_ExportSubContainer_String(ipCodeDecodedList, 0, sCodeAsString, 255, hr);
    sResultText := CONCAT('Code: ', sCodeAsString);
ELSE
    // Check for succeeded return codes or add specific error handling
    CASE hr OF
        S_FALSE:
            sResultText := 'No code found...';

        S_WATCHDOG_TIMEOUT:
            sResultText := 'Cancelled by watchdog...';
    ELSE
        // Extract error-code from HRESULT & react accordingly
        nReturnCode := DINT_TO_DWORD(hr) AND 16#FFF;
        sReturnCode := DWORD_TO_HEXSTR(nReturnCode, 3, FALSE);
        sResultText := CONCAT('Returncode ', sReturnCode);
    END_CASE
END_IF

// Draw result image
hr := F_VN_PutTextExp(sResultText, ipImageRes, 50, 100, ETcVnFontType.TCVN_FT_HERSHEY_PLAIN, 4,
aColorGreen, 3, TCVN_LT_4_CONNECTED, FALSE, S_OK);
hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, hr);
```

HINWEIS

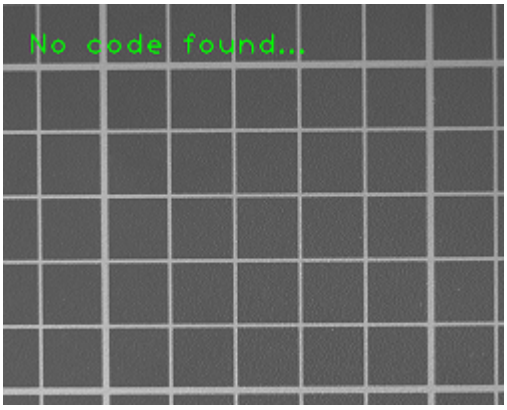
hr reservieren

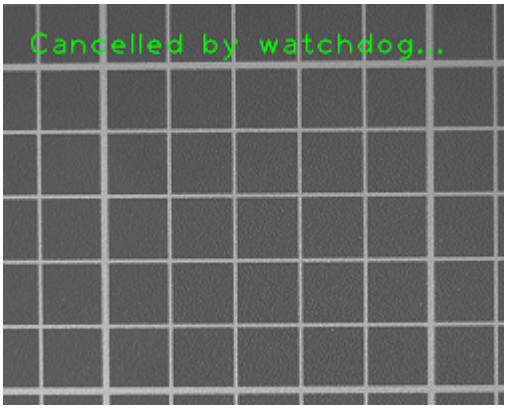
Achten Sie darauf, die Variable hr zwischen dem Aufruf der Code Reading Funktion und der Auswertung nicht als Rückgabewert einer anderen Funktion zu benutzen.

Ergebnisse

Wenn wie erwartet ein Code gefunden wird, ist `hr = S_OK` und das Ergebnis kann ausgegeben werden.

Wenn der erwartete Code nicht gefunden werden kann, entspricht der Return-Code `S_FALSE` und bei Abbruch durch einen Watchdog [▶ 156] ist er `16#256`. Eine entsprechende Meldung kann ausgegeben werden.





Wenn die Funktion nicht korrekt ausgeführt wird, kann der Fehler-Code ausgegeben werden. Dies kommt im gegebenen Beispiel nicht vor und wird bei Code Reading Funktionen nur durch falsche Benutzung hervorgerufen.

Ähnliche Beispiele

- Data Matrix Code Reading [▶ 2848]
- EAN-13 Barcode Reading [▶ 2845]
- Pharma-Code Reading [▶ 2850]
- QR-Code Reading [▶ 2852]

8.1.4 Contour Analysis

8.1.4.1 Match Contours 1vsN (manuelle Formen)

In diesem Beispiel werden Konturen mit Hilfe der Funktion F_VN_MatchContours1vsN [▶ 1014] mit Referenz-Konturen verglichen. Dabei sind

- die Referenz-Konturen idealisierte, manuell erstellte Polygone
- und die anderen Konturen werden aus einem Bild extrahiert.

Erläuterung

Die Funktion F_VN_MatchContours [▶ 1013] vergleicht zwei gegebene Konturen mit Hilfe der Hu Momente. Die Funktion F_VN_MatchContours1vsN [▶ 1014] stellt eine Erweiterung dar, mit der eine Kontur gleichzeitig mit mehreren anderen Konturen verglichen werden kann. Dies wird in diesem Beispiel benutzt, um Werkstücke bezüglich vorgegebener Referenzformen zu klassifizieren.

Die Referenzformen sind idealisierte, manuell erstellte Polygone eines gleichschenkligen Dreiecks, eines Rechtecks/Quadrats und eines Trapezes. Da bei den Polygonen nur die Eckpunkte angegeben wurden, unterscheiden sich die Referenzformen von den Testkonturen in der Anzahl der zu beschreibenden Konturpunkte stark. Weiterhin unterscheiden sie sich in Lage, Orientierung, Größe und dem Grad der Störungen innerhalb der Kontur.

Um die Form der Bauteile aus dem Bild zu extrahieren, werden mit Hilfe der Funktion F_VN_DetectBlobs [▶ 1207] Konturen gesucht, die eine gewisse Mindestgröße haben. Die gefundenen Konturen werden daraufhin für das beschriebene Matching genutzt.

Eingangsparameter

Neben den zu vergleichenden Konturen, muss der Funktion nur ein Parameter übergeben werden, der die Berechnungsmethode beschreibt, mit der aus den Hu Momenten das Ungleichheitsmaß bestimmt wird (**eComparisonMethod** vom Typ ETcVnContoursMatchComparisonMethod [▶ 204]). Die Methode `TCVN_CMCM_CONTOURS_MATCH_I1` berechnet die Summe über die Differenzen der Kehrwerte der individuellen Charakteristiken, während die Methode `TCVN_CMCM_CONTOURS_MATCH_I2` die Summe über die Differenzen der reinen Charakteristiken bildet. Im Gegensatz zu den ersten beiden Methoden berechnet die dritte Variante `TCVN_CMCM_CONTOURS_MATCH_I3` nur die maximale Differenz der individuellen Charakteristiken. Je nach Anwendungsfall kann eine andere der drei Methoden am besten geeignet sein für den Vergleich zweier Konturen.

Variablen

> **Datentyp nur in diesem Beispiel**
>
> Der Datentyp CustomTcVnArray4_Point2_DINT ist nicht in TwinCAT Vision enthalten, sondern wurde explizit für dieses Beispiel erstellt. Die Implementierung finden Sie im Download dieses Beispiels.

```
// Indexes representing the different shapes
TRIANGLE              :   USINT := 0;
RECTANGLE             :   USINT := 1;
TRAPEZOID             :   USINT := 2;

// Images
ipImageIn             :   ITcVnImage;
ipImageInDisp         :   ITcVnDisplayableImage;
ipImageRes            :   ITcVnImage;
ipImageResDisp        :   ITcVnDisplayableImage;

// Contours
ipContourList         :   ITcVnContainer;
ipContour             :   ITcVnContainer;
ipIterator            :   ITcVnForwardIterator;
aShapeArrays          :   ARRAY [0..2] OF CustomTcVnArray4_Point2_DINT :=
                              [[[0,0],[40,0],[20, LREAL_TO_DINT(SQRT(1200))], [0,0]],  // Triangle
                               [[0,0],[40,0],[40,40],[0,40]],                          // Rectangle
                               [[0,0],[40,0],[60,22],[20,22]]];                        // Trapezoid;
aShapes               :   ARRAY[0..2] OF ITcVnContainer;
```

```
ipShapes              :   ITcVnContainer;

// Matching
ipMatchIndexes        :   ITcVnContainer;
ipDissimilarities     :   ITcVnContainer;
aMatchIndexes         :   ARRAY [0..2] OF ULINT;
aDissimilarities      :   ARRAY [0..2] OF LREAL;

// Parameters
stBlobParams          :   TcVnParamsBlobDetection;
fThreshold            :   REAL := 170;
fMinArea              :   REAL := 10000;
fMaxDissimilarity     :   LREAL := 0.02;
eComparisonMethod     :   ETcVnContoursMatchComparisonMethod := TCVN_CMCM_CONTOURS_MATCH_I3;

// drawing
aColors               :   ARRAY[0..2] OF TcVnVector4_LREAL :=
                              [[0, 175, 0, 0], [0, 0, 255, 0], [255, 0, 0, 0]]; // green, blue, red
aColorWhite           :   TcVnVector4_LREAL := [255, 255, 255];
aColorBlack           :   TcVnVector4_LREAL := [0, 0, 0];
aTexts                :   ARRAY[0..2] OF STRING := ['Triangle', 'Rectangle', 'Trapezoid'];
sText                 :   STRING(255);
nTopLeftX             :   UDINT;
nTopLeftY             :   UDINT;

// Miscellaneous
aPixelValue           :   TcVnVector4_LREAL;
I                     :   USINT;
stBoundingRectangle   :   TcVnRectangle_UDINT;
aOffsets              :   ARRAY [0..2] OF TcVnPoint := [[20, 50], [20, 100], [20, 160]];
hr                    :   HRESULT;
```

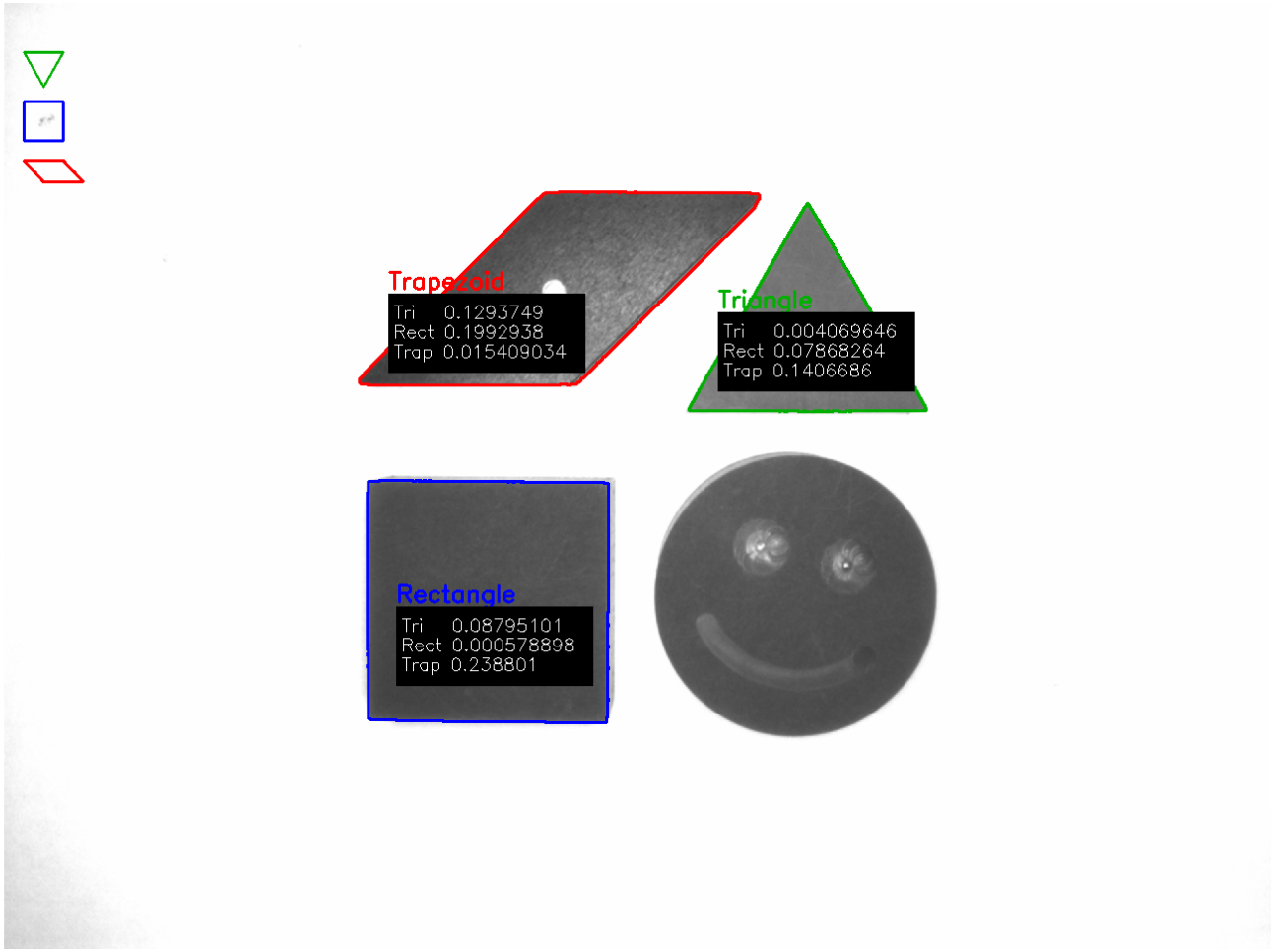
Code

```
// Fill manually defined shapes into one container
hr := F_VN_CreateContainer(ipShapes, ContainerType_Vector_Vector_TcVnPoint2_DINT, 0, hr);
FOR i:=TRIANGLE TO TRAPEZOID DO
    hr := F_VN_CreateContainerFromArray(ADR(aShapeArrays[i]), aShapes[i],
ContainerType_Vector_TcVnPoint2_DINT, 4, hr);
    hr := F_VN_InsertIntoContainer_ITcVnContainer(aShapes[i], ipShapes, i, hr);
END_FOR

// Prepare result image
hr := F_VN_ConvertColorSpace(ipImageIn, ipImageRes, TCVN_CST_GRAY_TO_RGB, hr);

// Check if background is light or dark
hr := F_VN_GetPixel(ipImageIn, aPixelValue, 50, 50, hr);
IF SUCCEEDED(hr) AND_THEN aPixelValue[0] < 128 THEN
    stBlobParams.eThresholdType := TCVN_TT_BINARY;
ELSE
    stBlobParams.eThresholdType := TCVN_TT_BINARY_INV;
END_IF

// Find contours in image
stBlobParams.bFilterByArea := TRUE;
stBlobParams.fMinArea := fMinArea;
stBlobParams.fMinThreshold := fThreshold;
hr := F_VN_DetectBlobs(ipImageIn, ipContourList, stBlobParams, hr);

// Iterate through all found contours
hr := F_VN_GetForwardIterator(ipContourList, ipIterator, hr);
WHILE hr = S_OK AND_THEN ipIterator.CheckIfEnd() <> S_OK DO
    hr := F_VN_GetContainer(ipIterator, ipContour, hr);
    hr := F_VN_IncrementIterator(ipIterator, hr);

    // Match the current contour with the reference shapes
    hr := F_VN_MatchContours1vsN(
        ipRefContour:= ipContour,
        ipContours:= ipShapes,
        ipMatchIndexes:= ipMatchIndexes,
        ipDissimilarities:= ipDissimilarities,
        fDissimilarityThreshold:= 100,
        eComparisonMethod:= eComparisonMethod,
        hrPrev:= hr
    );

    hr := F_VN_ExportContainer(ipMatchIndexes, ADR(aMatchIndexes), SIZEOF(aMatchIndexes), hr);
    hr := F_VN_ExportContainer(ipDissimilarities, ADR(aDissimilarities), SIZEOF(aDissimilarities),
hr);
```

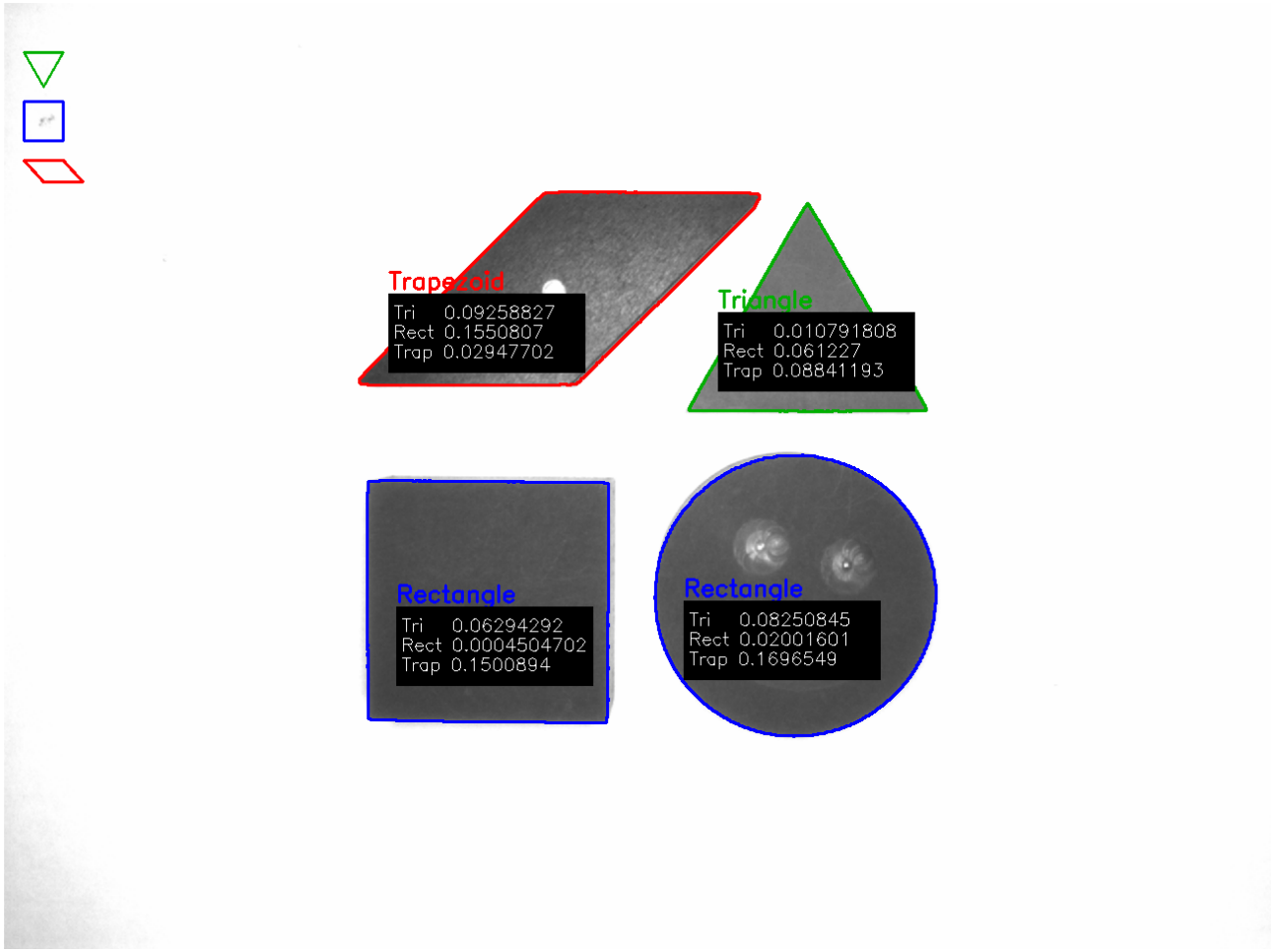
```
    // Draw matching results
    IF aDissimilarities[0] < fMaxDissimilarity THEN

        // Calculate position of object
        hr := F_VN_UprightBoundingRectangle(ipContour, stBoundingRectangle, hr);
        nTopLeftX := LREAL_TO_UDINT(stBoundingRectangle.nX + 30);
        nTopLeftY := LREAL_TO_UDINT(stBoundingRectangle.nY + (stBoundingRectangle.nHeight / 2));

        // Draw matching result
        hr := F_VN_DrawContours(ipContour, -1, ipImageRes, aColors[aMatchIndexes[0]], 5, hr);
        hr := F_VN_PutTextExp(aTexts[aMatchIndexes[0]], ipImageRes, nTopLeftX, nTopLeftY,
TCVN_FT_HERSHEY_SIMPLEX, 0.8, aColors[aMatchIndexes[0]], 2, TCVN_LT_8_CONNECTED, FALSE, hr);
        hr := F_VN_DrawRectangle(nTopLeftX, nTopLeftY + 5, nTopLeftX + 200, nTopLeftY + 85,
ipImageRes, aColorBlack, -1, hr);
        // Draw all dissimilarity values
        FOR i:=0 TO 2 DO
            sText := CONCAT(LEFT(aTexts[aMatchIndexes[i]], 4), CONCAT(' ',
REAL_TO_STRING(LREAL_TO_REAL(aDissimilarities[i]))));
            hr := F_VN_PutTextExp(sText, ipImageRes, nTopLeftX + 5, nTopLeftY + 30 + 20*i,
TCVN_FT_HERSHEY_SIMPLEX, 0.6, aColorWhite, 1, TCVN_LT_8_CONNECTED, FALSE, hr);
        END_FOR

    END_IF
END_WHILE

// Draw reference shapes
FOR i:=TRIANGLE TO TRAPEZOID DO
    hr := F_VN_DrawContoursExp(aShapes[i], -1, ipImageRes, aColors[i], 2, TCVN_LT_8_CONNECTED, 0, 0,
aOffsets[i], hr);
END_FOR
hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, hr);
hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, hr);
```

Ergebnisse

Das Ergebnisbild zeigt die Referenzformen (linker Rand), sowie diejenigen gefundenen Konturen, deren Ungleichheitsmaß bezogen auf alle Referenzformen einen Maximalwert nicht überschreitet. Weiterhin wird visualisiert, welcher Referenzform die gefundene Kontur am meisten ähnelt. Ein entsprechender Beschreibungstext (inklusive Ungleichheitsmaße für jede Referenzform) wird über die jeweilige Kontur gezeichnet. Für die vorgegebenen Parameter sieht das Ergebnis für das Bild MatchContours1.bmp wie folgt aus:



Es ist zu erkennen, dass der mit Abstand niedrigste Wert jeweils auf die korrekte Referenzkontur hinweist. Wird nun der Parameter eComparisonMethod von `TCVN_CMCM_CONTOURS_MATCH_I3` auf `TCVN_CMCM_CONTOURS_MATCH_I2` verändert, erkennt man, dass die Ergebnisse sich stark verändern. Dies führt auch dazu, dass der Parameter für die maximal akzeptierte Ungleichheit angepasst werden muss:



Die neuen Werte lassen eine deutlich weniger eindeutige Zuordnung zu entsprechenden Referenzformen zu, außerdem wird das runde Bauteil fälschlicher Weise der Referenzform „Rectangle“ zugeordnet. Diese Berechnungsmethode ist für dieses Beispiel also offenbar weniger gut geeignet.

Ähnliche Beispiele

- Match Contours (extrahierte Formen) [▶ 2861]

8.1.4.2 Match Contours (extrahierte Formen)

In diesem Beispiel werden zwei Konturen mit Hilfe der Funktion F_VN_MatchContours [▶ 1013] miteinander verglichen.

Erläuterung

Die Funktion F_VN_MatchContours [▶ 1013] vergleicht zwei gegebene Konturen mit Hilfe der Hu Momente. In diesem Beispiel wird die Funktion benutzt, um ein vorgegebenes Bauteil in einer neuen Situation wiederzuerkennen. Hierzu wird in einem Referenzbild ein Bauteil ausgewählt, dessen Kontur abgespeichert und anschließend mit den Konturen aus einem anderen Bild verglichen. Auf diese Weise werden die Bauteile, die eine gewisse Ungleichheit gegenüber dem Referenzbauteil unterschreiten markiert.

Eingangsparameter

Neben den zu vergleichenden Konturen, muss der Funktion nur ein Parameter übergeben werden, der die Berechnungsmethode beschreibt, mit der aus den Hu Momenten das Ungleichheitsmaß bestimmt wird (**eComparisonMethod** vom Typ ETcVnContoursMatchComparisonMethod [▶ 204]). Die Methode TCVN_CMCM_CONTOURS_MATCH_I1 berechnet die Summe über die Differenzen der Kehrwerte der individuellen Charakteristiken, während die Methode TCVN_CMCM_CONTOURS_MATCH_I2 die Summe über die Differenzen der reinen Charakteristiken bildet. Im Gegensatz zu den ersten beiden Methoden

berechnet die dritte Variante TCVN_CMCM_CONTOURS_MATCH_I3 nur die maximale Differenz der individuellen Charakteristiken. Je nach Anwendungsfall kann eine andere der drei Methoden am besten geeignet sein für den Vergleich zweier Konturen.

Variablen

```
hr                      :   HRESULT;

ipImageIn               :   ITcVnImage;
ipImageInDisp           :   ITcVnDisplayableImage;

ipImageRes              :   ITcVnImage;
ipImageResDisp          :   ITcVnDisplayableImage;
ipIterator              :   ITcVnForwardIterator;

// result
ipContourList           :   ITcVnContainer;
ipContourReference      :   ITcVnContainer;
ipContourCheck          :   ITcVnContainer;
fBestDissimilarity      :   LREAL;
fDissimilarity          :   LREAL;

// parameters
fThreshold              :   REAL := 170;
fMinArea                :   REAL := 10000;
fMaxDissimilarity       :   LREAL := 0.01;
eComparisonMethod       :   ETcVnContoursMatchComparisonMethod := TCVN_CMCM_CONTOURS_MATCH_I3;

// drawing
aColorGreen             :   TcVnVector4_LREAL := [0, 175, 0];
aColorBlue              :   TcVnVector4_LREAL := [0, 0, 255];
aColorRed               :   TcVnVector4_LREAL := [255, 0, 0];
aColorRes               :   TcVnVector4_LREAL;
sText                   :   STRING(255);
sTextReference          :   STRING(255) := 'Reference contour';
sTextCheck              :   STRING(255) := 'Check dissimilarity';

// other
bDarkBackground         :   BOOL;
nContours               :   ULINT;
nCounter                :   UINT := 0;
aPixelValue             :   TcVnVector4_LREAL;
stBoundingRectangle     :   TcVnRectangle_UDINT;
stParams                :   TcVnParamsBlobDetection;
```

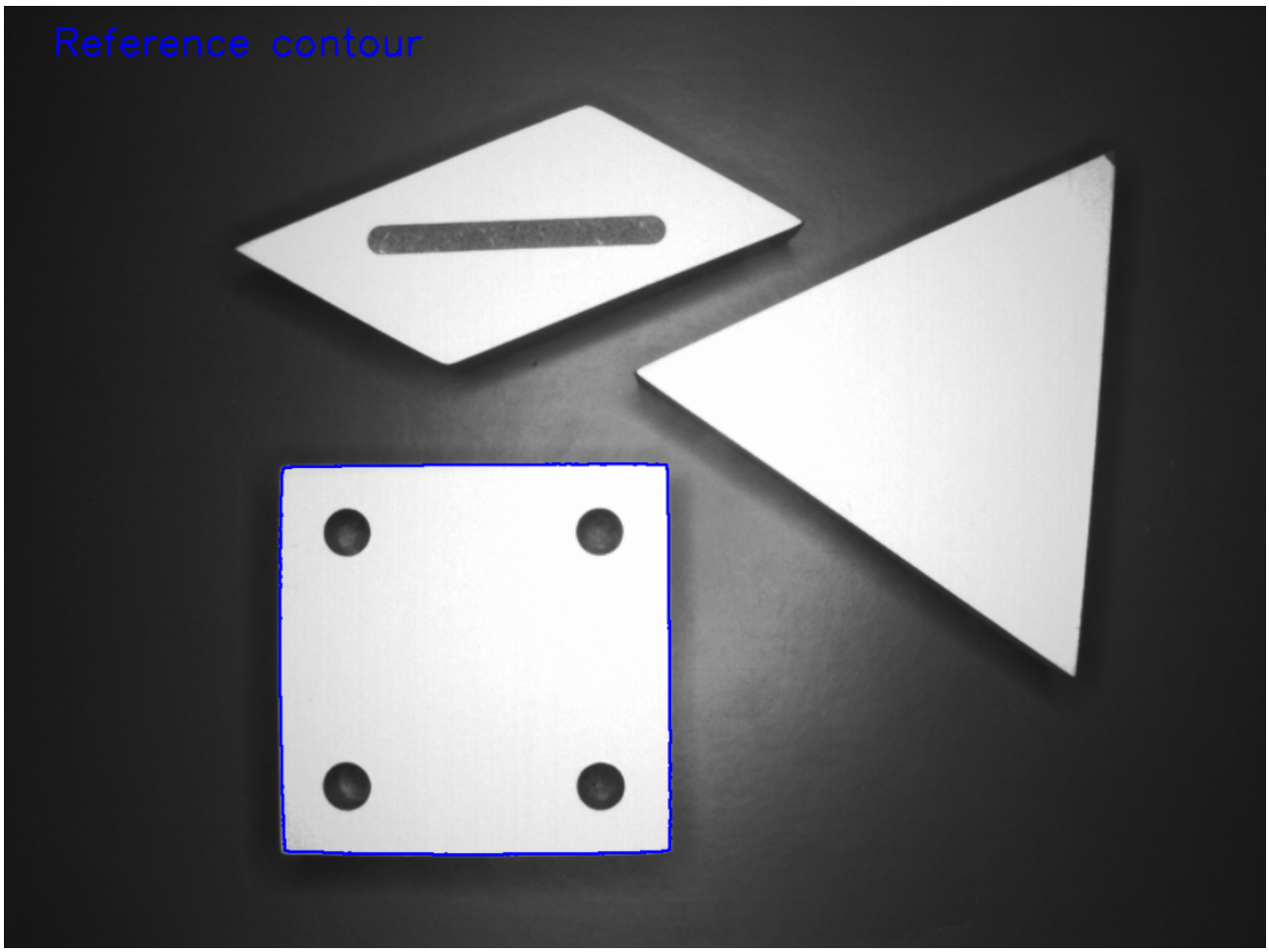
Code

```
// Prepare result image
hr := F_VN_ConvertColorSpace(ipImageIn, ipImageRes, TCVN_CST_GRAY_TO_RGB, hr);

// Check if background is dark in order to identify reference image
hr := F_VN_GetPixel(ipImageIn, aPixelValue, 50, 50, hr);
bDarkBackground := SUCCEEDED(hr) AND_THEN aPixelValue[0] < 128;

IF bDarkBackground THEN
    stParams.eThresholdType := TCVN_TT_BINARY;
ELSE
    stParams.eThresholdType := TCVN_TT_BINARY_INV;
END_IF

// Find contours in image
hr := F_VN_DetectBlobs(ipImageIn, ipContourList, stParams, hr);

// Distinguish reference and test image
IF bDarkBackground THEN

    // Select one of the found contours as reference for matching
    hr := F_VN_GetNumberOfElements(ipContourList, nContours, hr);
    hr := F_VN_GetAt_ITcVnContainer(ipContourList, ipContourReference, nCounter MOD nContours, hr);

    // Draw selected contour and text
    hr := F_VN_PutTextExp(sTextReference, ipImageRes, 50, 50, TCVN_FT_HERSHEY_SIMPLEX, 1.3,
aColorBlue, 2, TCVN_LT_8_CONNECTED, FALSE, hr);
    hr := F_VN_DrawContours(ipContourReference, -1, ipImageRes, aColorBlue, 5, hr);

    nCounter := nCounter + 1;

ELSE
```

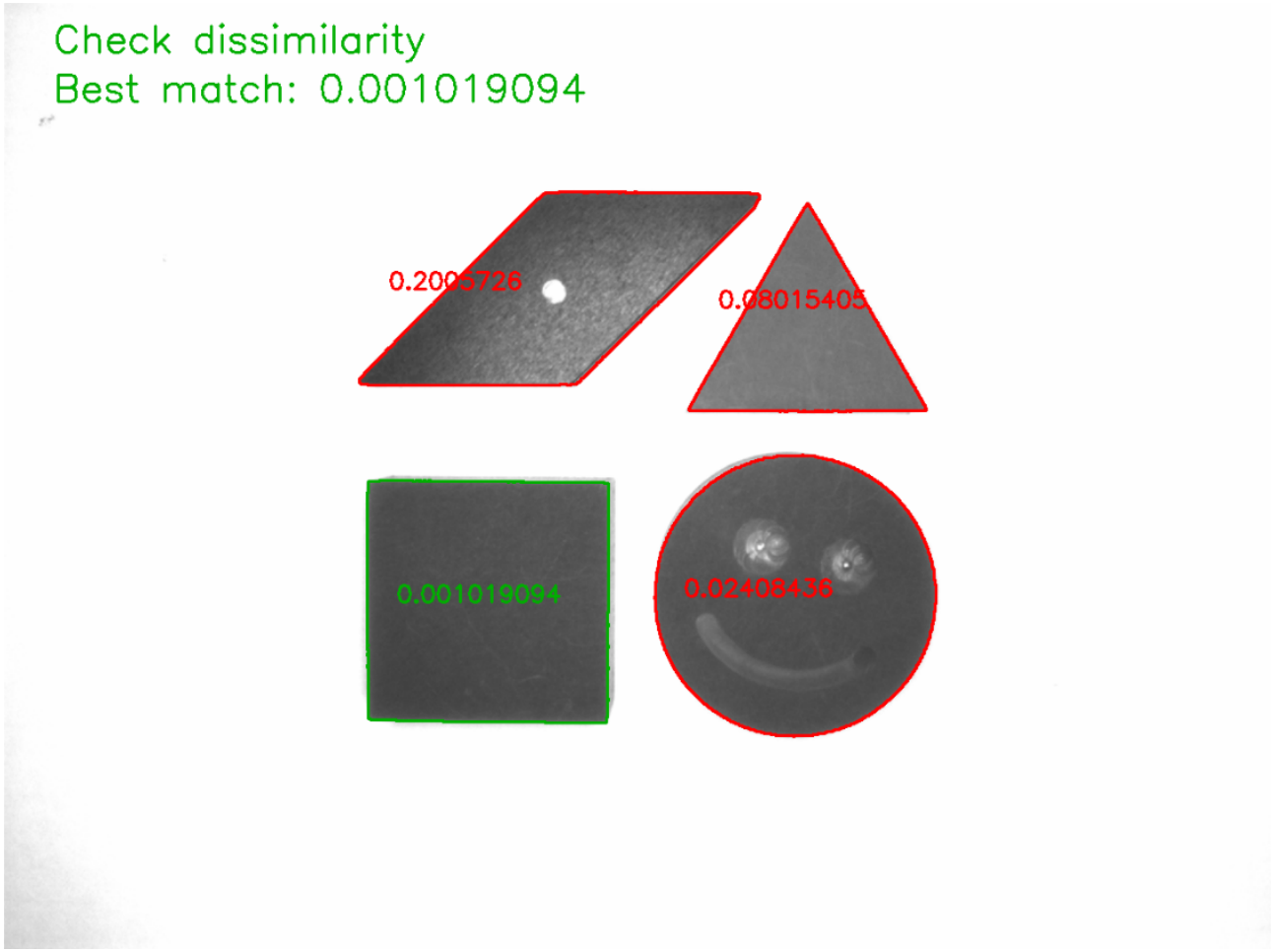
```
    fBestDissimilarity := 10E300;

    // Iterate through all found contours
    hr := F_VN_GetForwardIterator(ipContourList, ipIterator, hr);
    WHILE hr = S_OK AND_THEN ipIterator.CheckIfEnd() <> S_OK DO
        hr := F_VN_GetContainer(ipIterator, ipContourCheck, hr);
        hr := F_VN_IncrementIterator(ipIterator, hr);

        // Match the current contour with the selected reference contour
        hr := F_VN_MatchContours(ipContourReference, ipContourCheck, eComparisonMethod,
fDissimilarity, hr);

        // Save best result
        IF fBestDissimilarity > fDissimilarity THEN
            fBestDissimilarity := fDissimilarity;
        END_IF

        // Choose action depending on the dissimilarity of both contours
        IF SUCCEEDED(hr) AND_THEN fDissimilarity < fMaxDissimilarity THEN
            aColorRes := aColorGreen;
        ELSE
            aColorRes := aColorRed;
        END_IF

        // Draw matching results
        sText := REAL_TO_STRING(LREAL_TO_REAL(fDissimilarity));
        hr := F_VN_UprightBoundingRectangle(ipContourCheck, stBoundingRectangle, hr);
        hr := F_VN_DrawContours(ipContourCheck, -1, ipImageRes, aColorRes, 5, hr);
        hr := F_VN_PutTextExp(sText, ipImageRes, LREAL_TO_UDINT(stBoundingRectangle.nX + 30),
LREAL_TO_UDINT(stBoundingRectangle.nY + (stBoundingRectangle.nHeight / 2)), TCVN_FT_HERSHEY_SIMPLEX,
0.8, aColorRes, 2, TCVN_LT_8_CONNECTED, FALSE, hr);

    END_WHILE

    // Draw text
    hr := F_VN_PutTextExp(sTextCheck, ipImageRes, 50, 50, TCVN_FT_HERSHEY_SIMPLEX, 1.3, aColorGreen,
2, TCVN_LT_8_CONNECTED, FALSE, hr);
    sText := CONCAT('Best match: ', REAL_TO_STRING(LREAL_TO_REAL(fBestDissimilarity)));
    hr := F_VN_PutTextExp(sText, ipImageRes, 50, 100, TCVN_FT_HERSHEY_SIMPLEX, 1.3, aColorGreen, 2,
TCVN_LT_8_CONNECTED, FALSE, hr);

END_IF

hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, hr);
hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, hr);
```

Ergebnisse

Auf dem Referenzbild wird die aktuell ausgewählte Referenzkontur in blau markiert:

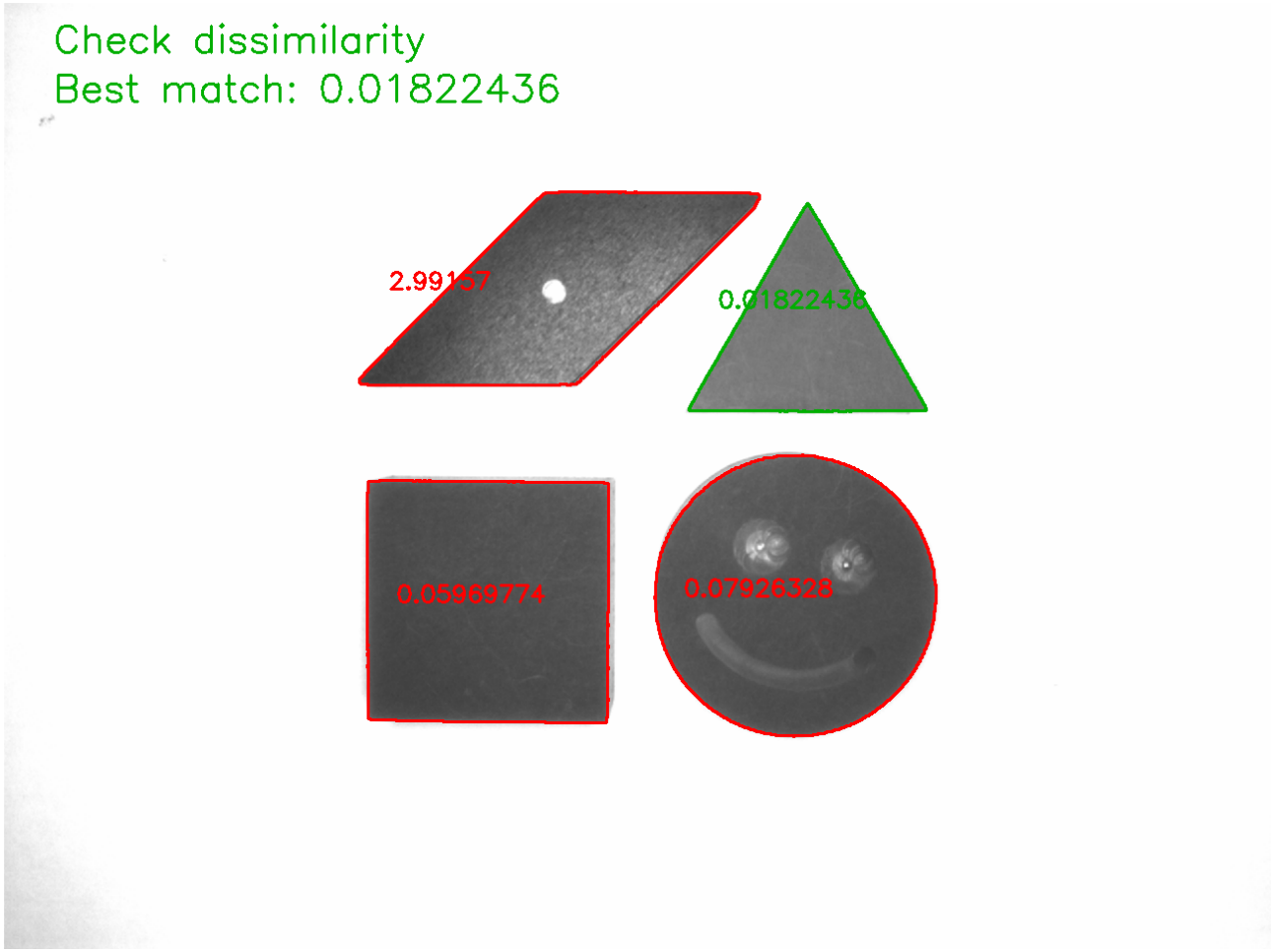


Man sieht, dass die Kontur durch die Linsenverzerrung leicht gewölbt ist und einige kleinere Störungen aufweist. Im Testbild werden Konturen, deren Ungleichheitsmaß gegenüber der Referenzkontur den eingestellten Schwellwert unterschreitet, in Grün dargestellt - die übrigen Konturen in Rot. Außerdem wird über alle Konturen der berechnete Ungleichheitswert gezeichnet. Im Folgenden ist das Ergebnis zum Eingabebild *MatchContours1.bmp* bei Auswahl des Rechtecks als Referenzbauteil abgebildet:



Es ist zu erkennen, dass das rechteckige Bauteil mit Abstand den kleinsten Ungleichheitswert gegenüber der Referenzkontur hat. Ändert man nun den Parameter eComparisonMethod von TCVN_CMCM_CONTOURS_MATCH_I3 auf TCVN_CMCM_CONTOURS_MATCH_I2, ändern sich entsprechend die berechneten Ungleichheitswerte:

In diesem Fall ist das Ergebnis noch eindeutiger, allerdings trifft das nicht auf alle Formen zu. Wenn das Dreieck als Referenzbauteil ausgewählt ist, wird das Ergebnis mit dieser Einstellung deutlich weniger eindeutig als bei der ursprünglichen Einstellung:



Dieses Ergebnis macht deutlich, dass je nach Anwendung eine andere Berechnungsmethode die besten Resultate liefern kann.

Ähnliches Beispiel

- Match Contours 1vsN (manuelle Formen) [▶ 2857]

8.1.5 Image Analysis

8.1.5.1 Object Detection

8.1.5.1.1 Blob Detection mit Watchdog Überwachung

In diesem Beispiel werden

- über die Funktion F_VN_DetectBlobs [▶ 1207] runde Strukturen in Bildern gesucht
- und die Ausführungszeit der Funktion über einen sogenannten Watchdog überwacht.

Erläuterung

Die Funktion F_VN_DetectBlobs [▶ 1207] erleichtert das Suchen gleichartiger Strukturen im Bild. Dazu kombiniert sie das Setzen von Schwellwerten mit dem Suchen und Filtern nach Konturen in einem Aufruf. In diesem Beispiel ist die Funktion so parametriert, dass sie alle runden Strukturen in den Beispielbildern zurückgibt.

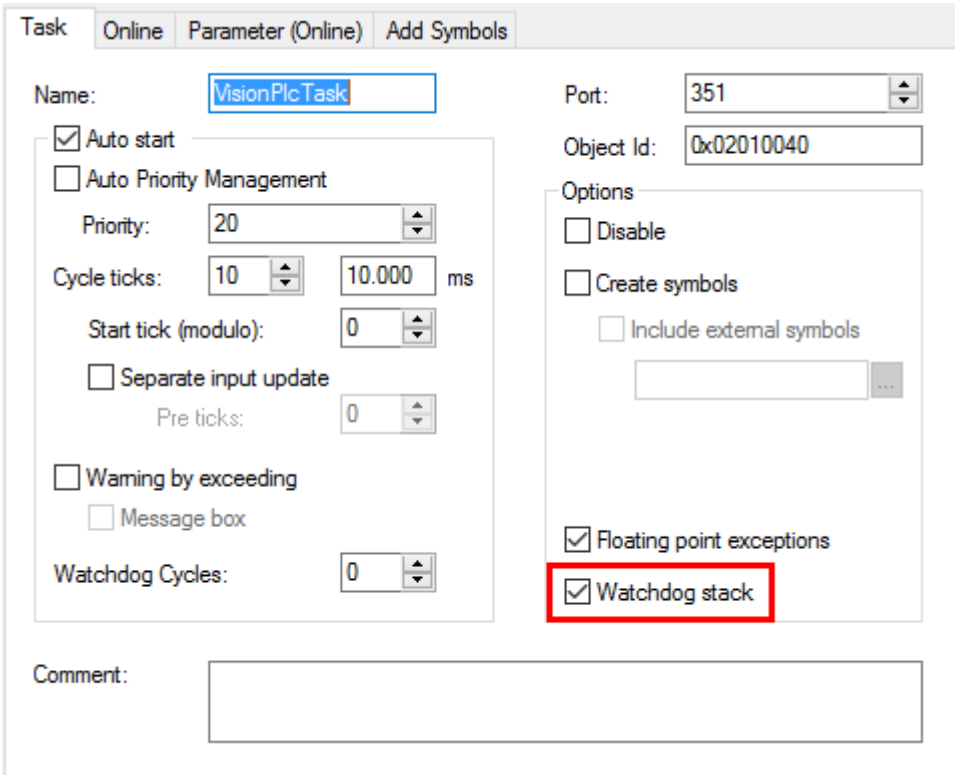
Durch äußere Einflüsse können die tatsächlichen Bildinhalte stark von den erwarteten Bildinhalten abweichen. Bei Bildverarbeitungsalgorithmen, deren Ausführungszeit vom Bildinhalt abhängt, kann dies zu Zykluszeitüberschreitungen und damit zu ungewünschten Nebeneffekten führen. Um dies zu verhindern gibt

es einen Watchdog. Dieser kann Funktionen, die mit dem Satz *„Can return partial results when canceled by Watchdog.“* in der Api gekennzeichnet sind, nach einer vorgegebenen Zeitspanne abbrechen. Die bis zu dem Zeitpunkt ermittelten Ergebnisse können weiterverwendet werden.

Zu den überwachbaren Funktionen gehört auch F_VN_DetectBlobs. Durch Veränderung der zulässigen Ausführungszeit `tStop` im Watchdog-Start F_VN_StartRelWatchdog [▶ 1022] lässt sich sehr gut ein Funktionsabbruch und weiterarbeiten mit Teilergebnissen im Vergleich zur vollständigen Ausführung von F_VN_DetectBlobs beobachten. Über F_VN_StopWatchdog [▶ 1024] wird der Watchdog wieder gestoppt. Dabei gibt ein positiver Rückgabewert in `tRest` die noch verbleibende Zeit an, während ein negativer Wert in `tRest` die zum Abbruch zusätzlich benötigte Zeit angibt.

Task Konfiguration

Die Watchdog-Überwachung [▶ 156] muss in der ausführenden SPS Task aktiviert werden. Andernfalls werden die Watchdog-Funktionsaufrufe in der SPS ignoriert und die Start-Watchdog-Funktion gibt den Rückgabewert 16#71A „No Interface“ zurück.



Variablen

```
hr                  :   HRESULT;
hrFunc              :   HRESULT;

ipImageIn           :   ITcVnImage;
ipImageInDisp       :   ITcVnDisplayableImage;

ipImageRes          :   ITcVnImage;
ipImageResDisp      :   ITcVnDisplayableImage;

// Blob Detection
stBlobParams        :   TcVnParamsBlobDetection;
ipContourList       :   ITcVnContainer;

// Watchdog
hrWD                :   HRESULT;
tStop               :   DINT := 5000;
tRest               :   DINT;
nFraction           :   UDINT;

// Output
sText               :   STRING(255);

// Color
```

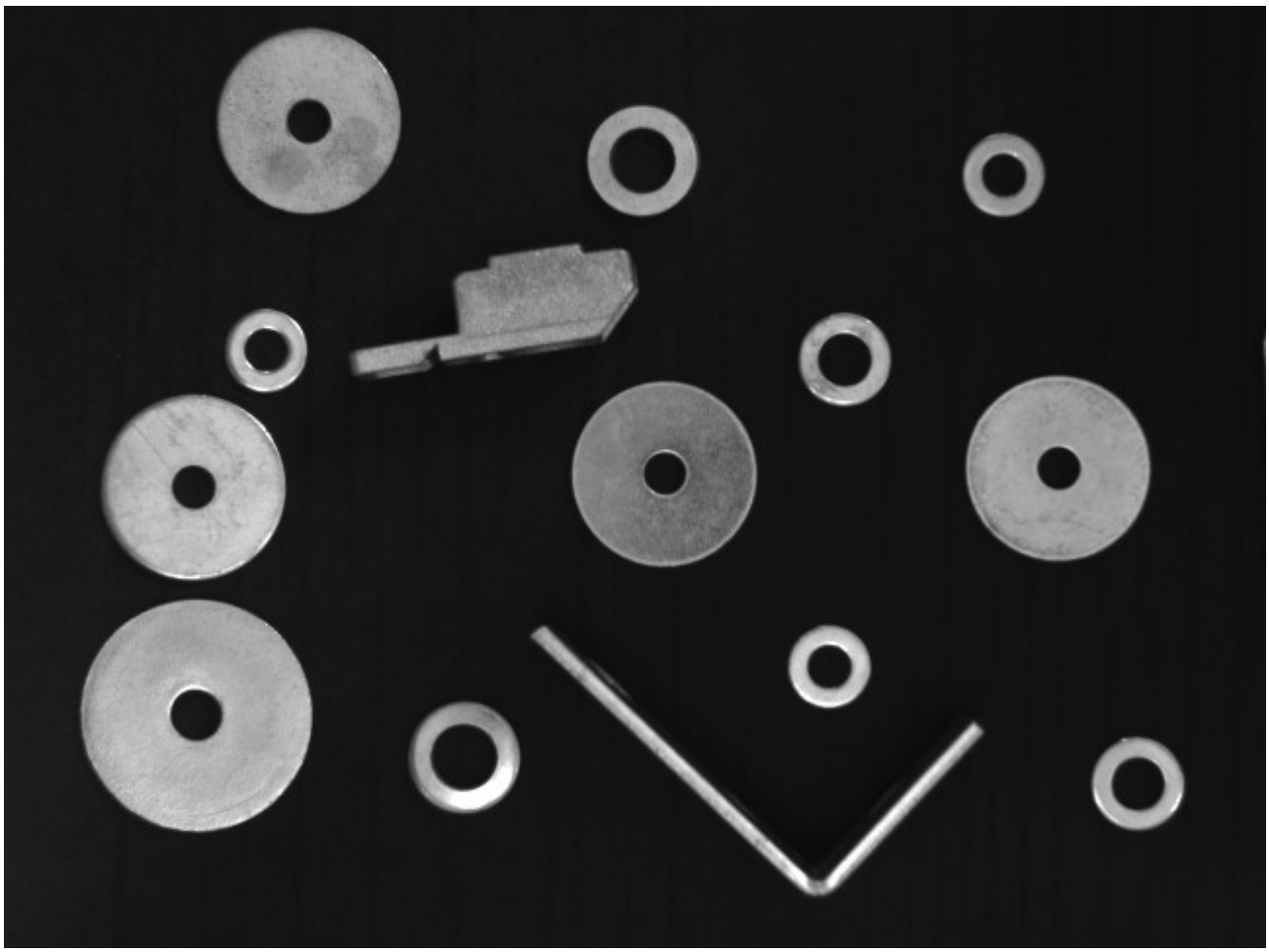
```
aColorGreen         :   TcVnVector4_LREAL := [0, 255, 0, 0];

// Image Infos
stPixelFormat       :   TcVnPixelFormat;
```

Code

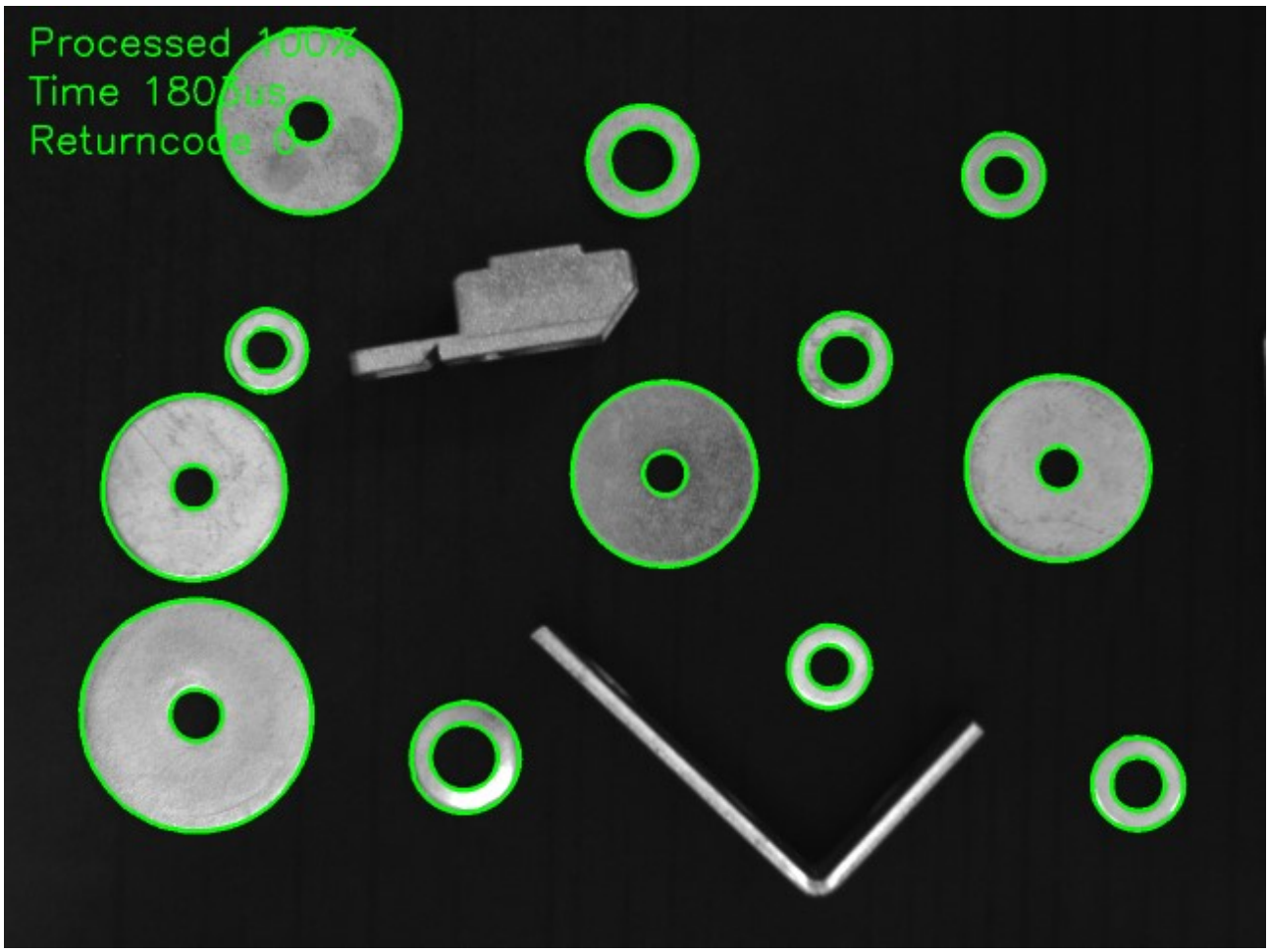
```
// Set Parameters
// ---------------------------------------------------------------
// - Set fMaxArea to 100000 detect all circles or set it to 10000
//   to detect only the small circles

     stBlobParams.bFilterByArea         := TRUE;
     stBlobParams.fMinArea              := 100;
     stBlobParams.fMaxArea              := 100000;

     stBlobParams.bFilterByCircularity  := TRUE;
     stBlobParams.fMinCircularity       := 0.80;

     stBlobParams.fMinThreshold         := 70;
     stBlobParams.fThresholdStep        := 0;

// Execute DetectBlobs-Function monitored by Watchdog-Function
// ---------------------------------------------------------------
// - Set the Watchdog stop time tStop from 5000us to 1000us to see
//   a Watchdog interrupt and that the interim results can be used

     hrWD := F_VN_StartRelWatchdog(tStop, S_OK);
         hrFunc := F_VN_DetectBlobs(
                                ipSrcImage      := ipImageIn,
                                ipBlobContours  := ipContourList,
                                stParams        := stBlobParams,
                                hrPrev          := hr);
     hrWD := F_VN_StopWatchdog(hrWD, tRest => tRest, nFractionProcessed => nFraction);

// Draw Result Image
// ---------------------------------------------------------------
hr := F_VN_DrawContours(ipContourList, -1, ipImageRes, aColorGreen, 3, hr);

sText := CONCAT(CONCAT('Processed ', UDINT_TO_STRING(nFraction)), '%');
hr := F_VN_PutTextExp(sText, ipImageRes, 25, 50, TCVN_FT_HERSHEY_SIMPLEX, 1.3, aColorGreen, 2,
TCVN_LT_8_CONNECTED, FALSE, hr);

sText := CONCAT(CONCAT('Time ', DINT_TO_STRING(tStop - tRest)), 'us');
hr := F_VN_PutTextExp(sText, ipImageRes, 25, 100, TCVN_FT_HERSHEY_SIMPLEX, 1.3, aColorGreen, 2,
TCVN_LT_8_CONNECTED, FALSE,hr);

sText := CONCAT('Returncode ', DINT_TO_STRING(hrFunc));
hr := F_VN_PutTextExp(sText, ipImageRes, 25, 150, TCVN_FT_HERSHEY_SIMPLEX, 1.3, aColorGreen, 2,
TCVN_LT_8_CONNECTED, FALSE,hr);
```

Ergebnisse

Das Eingangsbild

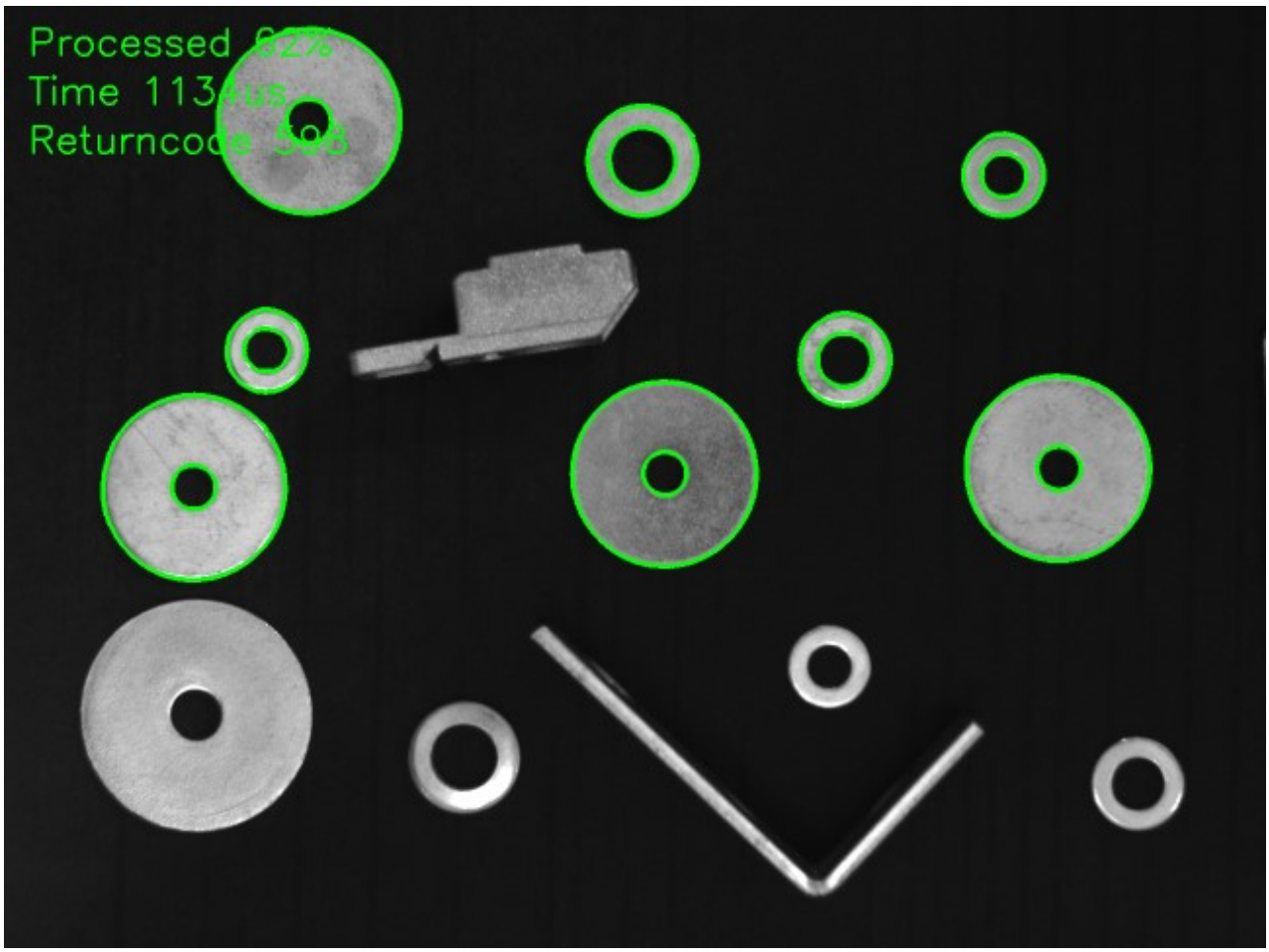


Alle Kreise im Bild gefunden, mit der Parametrierung des Beispielcodes.



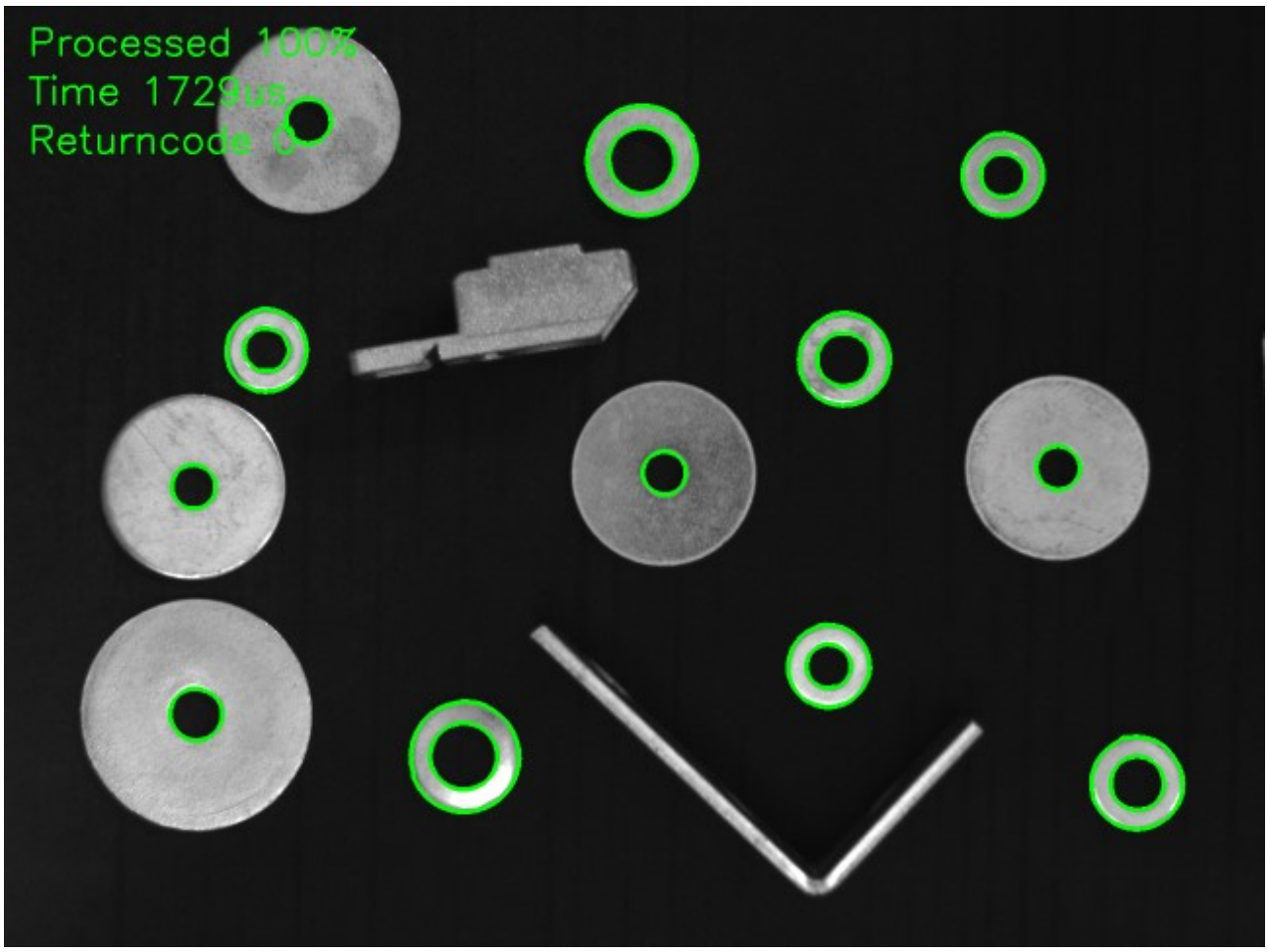
Einen Teil der Kreise im Bild gefunden, durch ein Auslösen des Watchdogs nach 1ms. Da auch der Abbruch Zeit kostet, ist die tatsächlich benötigte Zeit größer 1ms.

```
hrWD := F_VN_StartRelWatchdog(tStop:=1000, WATCHDOG_ACCUMULATION_TYPE_MEAN, S_OK);
```



Nur die kleinen Kreise im Bild gefunden durch Reduzierung der maximal umschließenden Fläche in Pixel.

```
stBlobParams.fMinArea := 100;
stBlobParams.fMaxArea := 10000;
```



Ähnliches Beispiel

- Find Contour anstelle der Blob Detection [▶ 2873]

8.1.5.1.2 Find Contour anstelle der Blob Detection

In diesem Beispiel wird eine alternative Lösung des Blob Detection Beispiels [▶ 2867] mit Hilfe von F_VN_FindContoursExp [▶ 1219] gezeigt.

Erläuterung

Die Funktion F_VN_DetectBlobs [▶ 1207] kapselt viele einzelne Bildverarbeitungsschritte in einem Funktionsaufruf. Die internen Verarbeitungsschritte können aber auch durch einzelne Funktionsaufrufe erfolgen. Dies kann das Setzen von Parametern erleichtern, da man die Zwischenergebnisse sieht. Daher werden in diesem Beispiel die gleichen Bilder wie in dem Blob Detection Beispiel [▶ 2867] verwendet und die gleichen runden Strukturen gesucht.

Variablen

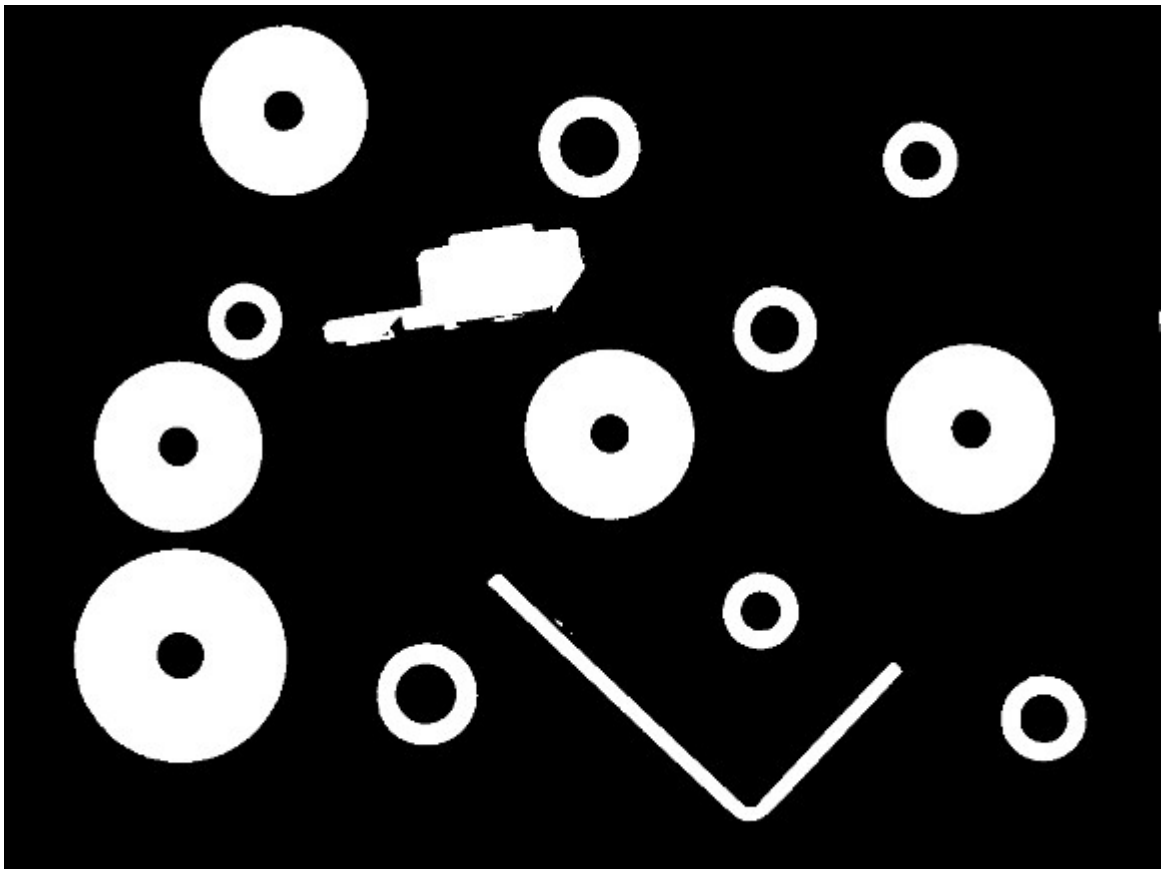
```
hr                    : HRESULT;

ipImageIn             : ITcVnImage;
ipImageInDisp         : ITcVnDisplayableImage;
ipImageWork           : ITcVnImage;
ipImageThresholdDisp  : ITcVnDisplayableImage;
ipImageRes            : ITcVnImage;
ipImageResDisp        : ITcVnDisplayableImage;

// Sample Specific Variables
ipContourList         : ITcVnContainer;
ipContourResultList   : ITcVnContainer;
ipContour             : ITcVnContainer;
ipIterator            : ITcVnForwardIterator;
```

```
aOffset                 : TcVnPoint;

fThreshold              : LREAL := 70;

fArea                   : LREAL;
fAreaMin                : LREAL := 100;
fAreaMax                : LREAL := 100000;

fCircularity            : LREAL;
fCircularityMin         : LREAL := 0.8;

// Image Infos
stPixelFormat           : TcVnPixelFormat;

// COLORS
aColorGreen             : TcVnVector4_LREAL := [0, 255, 0];
```

Code

Vorverarbeitung zur Bildsegmentierung

```
// Image Segementation
hr := F_VN_Threshold(ipImageIn, ipImageWork, fThreshold, 255, TCVN_TT_Binary, hr);
```



Kontursuche

```
// Find Contours
hr := F_VN_FindContoursExp(
ipSrcImage              := ipImageWork,
ipContours              := ipContourList,
eRetrievalMode          := ETcVnContourRetrievalmode.TCVN_CRM_LIST,
eApproximationMethod    := ETcVnContourApproximationMethod.TCVN_CAM_SIMPLE,
aOffset                 := aOffset,
hrPrev                  := hr);
```

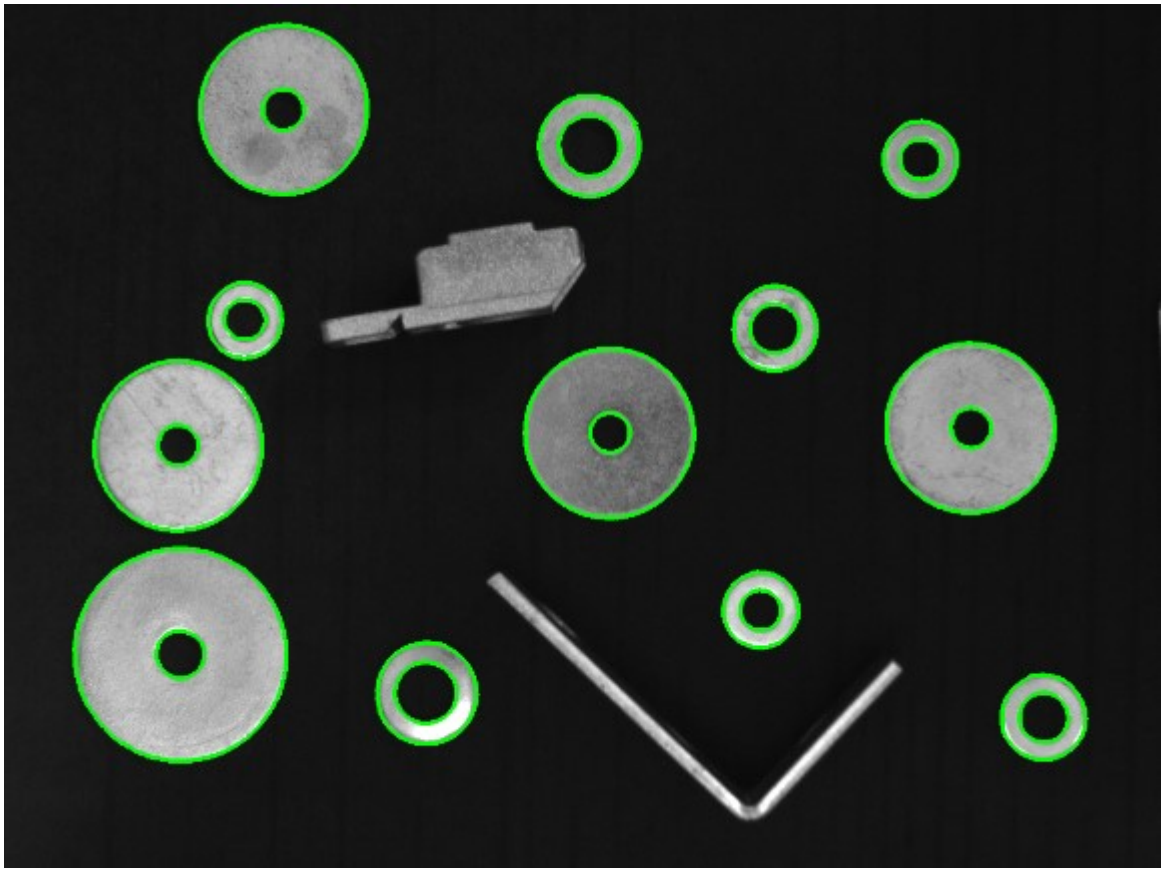
Filtern der Konturen nach der umschließenden Fläche in Pixeln und der Zirkularität

```
// Filter Contours
hr := F_VN_GetForwardIterator(ipContourList, ipIterator, hr);
hr := F_VN_CreateContainer(ipContourResultList, ContainerType_Vector_Vector_TcVnPoint2_DINT, 0, hr);

WHILE SUCCEEDED(hr) AND_THEN ipIterator.CheckIfEnd() <> S_OK DO
```

```
hr := F_VN_GetContainer(ipIterator, ipContour, hr);
hr := F_VN_IncrementIterator(ipIterator, hr);

// Filter by Area
hr := F_VN_ContourArea(ipContour, fArea, hr);

IF fArea > fAreaMin AND fArea < fAreaMax THEN

// Filter by Circularity
hr := F_VN_ContourCircularity(ipContour, fCircularity, hr);

IF fCircularity > fCircularityMin THEN
// Add contour to the result contour container
hr := F_VN_AppendToContainer_ITcVnContainer(ipContour, ipContourResultList, hr);

END_IF
END_IF
END_WHILE
```

Ergebnisausgabe

```
// Draw contours into the result image and display it
hr := F_VN_DrawContours(ipContourResultList, -1, ipImageRes, aColorGreen, 3, hr);
hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, hr);
```



Ähnliches Beispiel

- Blob Detection mit Watchdog Überwachung [▶ 2867]

8.1.5.1.3 Find Contour - Approximation Method

Dieses Beispiel beschäftigt sich mit den möglichen Annährungsmodi (**eApproximationMethod** vom Typ ETcVnContourApproximationMethod [▶ 199]) bei der Konturensuche.

Erläuterung

Die Funktionen F_VN_FindContours [▶ 1216], F_VN_FindContoursExp [▶ 1219] und F_VN_FindContourByHierarchy [▶ 1212] dienen zur Detektion von Konturen im Bild. Dabei werden die Konturen durch Punkte beschrieben. Je nach ausgewähltem Annäherungsmodus (**eApproximationMethod** vom Typ ETcVnContourApproximationMethod [▶ 199]) werden alle Punkte je Kontur oder eine vereinfachte Punktmenge zurückgegeben:

- TCVN_CAM_NONE
  alle Punkte der Kontur werden zurückgegeben
- TCVN_CAM_SIMPLE
  es erfolgt eine verlustfreie Komprimierung
- TCVN_CAM_TC89_L1 oder TCVN_CAM_TC89_KCOS
  wenden beide eine der Varianten des Teh-Chin-Kettenapproximations-Algorithmus an

Variablen

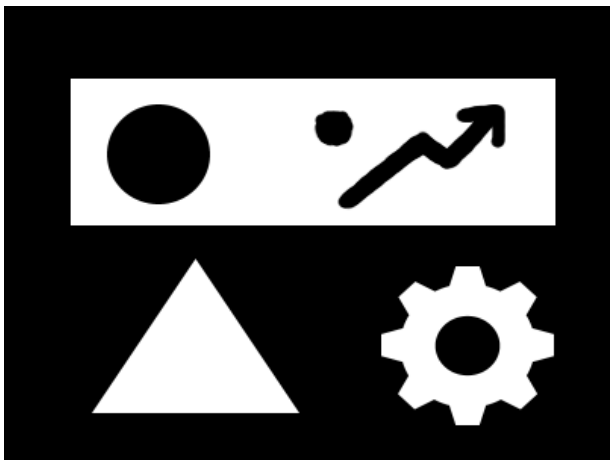
```
hr                   :   HRESULT;

ipImageIn            :   ITcVnImage;
ipImageInDisp        :   ITcVnDisplayableImage;

ipImageRes           :   ITcVnImage;
ipImageResDisp       :   ITcVnDisplayableImage;

// Sample Specific Variables
ipContourList        :   ITcVnContainer;
ipContour            :   ITcVnContainer;
ipIterator           :   ITcVnForwardIterator;
eRetrievalMode       :   ETcVnContourRetrievalMode := TCVN_CRM_LIST;
eApproximationMethod : ETcVnContourApproximationMethod := TCVN_CAM_SIMPLE;
aOffset              :   TcVnPoint;

// Colors
aColorRed            :   TcVnVector4_LREAL := [255, 0, 0];
```

Code

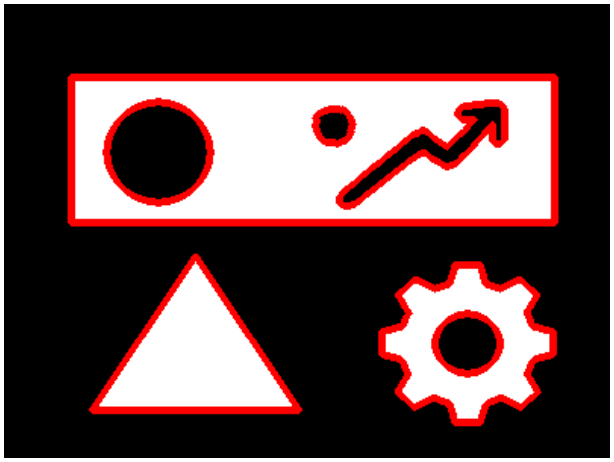
```
// Create Result Image
hr := F_VN_ConvertColorSpace(ipImageIn, ipImageRes, TCVN_CST_Gray_TO_RGB, hr);

// Find Contours and their Hierarchy
// ---------------------------------
hr := F_VN_FindContoursExp(
                ipSrcImage              :=   ipImageIn,
                ipContours              :=   ipContourList,
                eRetrievalMode          :=   eRetrievalMode,
                eApproximationMethod    :=   eApproximationMethod,
                aOffset                 :=   aOffset,
                hrPrev                  :=   hr);

// Draw the points of the contour
hr := F_VN_GetForwardIterator(ipContourList, ipIterator, hr);

WHILE SUCCEEDED(hr) AND_THEN ipIterator.CheckIfEnd() <> S_OK DO
   hr := F_VN_GetContainer(ipIterator, ipContour, hr);
   hr := F_VN_IncrementIterator(ipIterator, hr);
   hr := F_VN_DrawPoints(ipContour, ipImageRes, ETcVnDrawShape.TCVN_DS_CIRCLE, aColorRed, hr);
END_WHILE
```

Ergebnisse

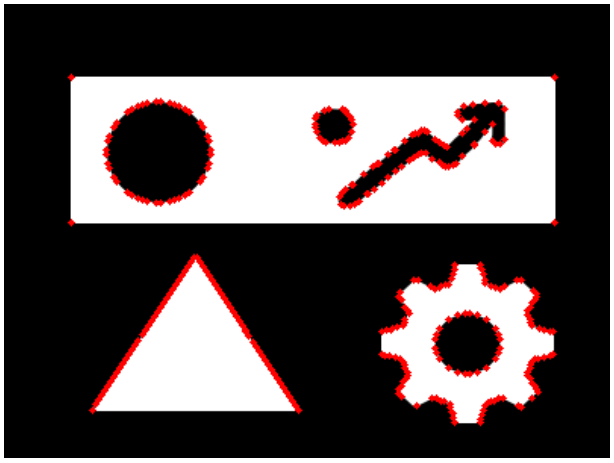
Das Eingangsbild



Das Ergebnisbild mit allen eingezeichneten Punkten nach ETcVnContourApproximationMethod [▶ 199].TCVN_CAM_NONE



Das Ergebnisbild mit allen eingezeichneten Punkten nach ETcVnContourApproximationMethod [▶ 199].TCVN_CAM_TC89_L1



Ähnliche Beispiele

- Find Contour - Hierarchy & Retrieval Mode [▶ 2878]
- Find Contour anstelle der Blob Detection [▶ 2873]

8.1.5.1.4 Find Contour - Hierarchy & Retrieval Mode

Dieses Beispiel zeigt den Retrieval-Mode (ETcVnContourRetrievalMode [▶ 202]) bei der Kontursuche (F_VN_FindContoursExp [▶ 1219] und F_VN_FindContourHierarchyExp [▶ 1212]) und wie sich dieser auf das Konturergebnis und die zurück gegebene Hierarchie auswirkt.

Erläuterung

Über den Konturabrufmodus (**eRetrievalMode** vom Typ ETcVnContourRetrievalMode [▶ 202]) kann angegeben werden, wie die Konturhierarchie bei der Kontursuche berücksichtigt werden soll. Dieses Beispiel visualisiert passend zum ausgewählten Modus die Ergebnisse.

Nähere Erläuterungen zur Hierarchiestruktur in F_VN_FindContourHierarchyExp [▶ 1212].

Variablen

```
hr                    :   HRESULT;

ipImageIn             :   ITcVnImage;
ipImageInDisp         :   ITcVnDisplayableImage;

ipImageRes            :   ITcVnImage;
ipImageResDisp        :   ITcVnDisplayableImage;

ipImageHierarchy      :   ITcVnImage;
ipImageHierarchyDisp  :   ITcVnDisplayableImage;

// Sample Specific Variables
ipContourList         :   ITcVnContainer;
ipContour             :   ITcVnContainer;
ipHierarchyList       :   ITcVnContainer;
aHierarchy            :   TcVnVector4_DINT;
aHierarchyTree        :   TcVnVector4_DINT;
eRetrievalMode        :   ETcVnContourRetrievalMode := TCVN_CRM_LIST;
eApproximationMethod  :   ETcVnContourApproximationMethod := TCVN_CAM_SIMPLE;
aOffset               :   TcVnPoint;

nNumOfElem            :   ULINT;
nParents              :   UINT;
i                     :   ULINT;
j                     :   ULINT;
stRect                :   TcVnRectangle_UDINT;
sText                 :   STRING(80);

// Colors
aColorWhite           :   TcVnVector4_LREAL := [255, 255, 255];
aColorBlack           :   TcVnVector4_LREAL := [0, 0, 0];
aColorList            :   ARRAY[0..4] OF TcVnVector4_LREAL := [ [255,0,0], [0,255,0], [0,0,255], [200
,200,0], [200,0,200]];
```

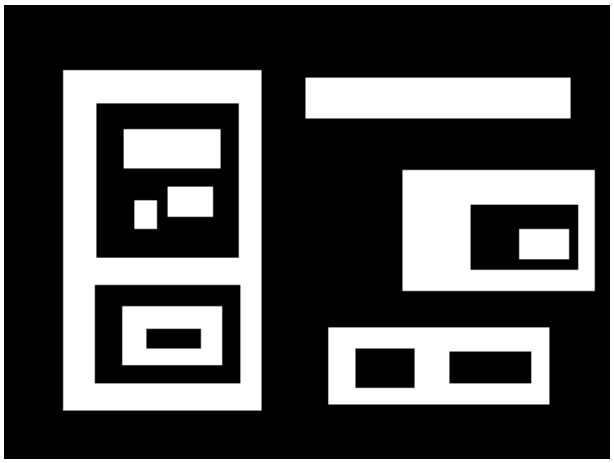
Code

```
// Create Result Image
hr := F_VN_ConvertColorSpace(ipImageIn, ipImageRes, TCVN_CST_Gray_TO_RGB, hr);
hr := F_VN_CreateImage(ipImageHierarchy, 130, 200, EtcVnElementType.TCVN_ET_USINT, 3, hr);
hr := F_VN_SetPixels(ipImageHierarchy, aColorWhite, hr);

// Find Contours and their Hierarchy
// ---------------------------------
hr := F_VN_FindContourHierarchyExp(
                ipSrcImage              :=   ipImageIn,
                ipContours              :=   ipContourList,
                ipHierarchy             :=   ipHierarchyList,
                eRetrievalMode          :=   eRetrievalMode,
                eApproximationMethod    :=   eApproximationMethod,
                aOffset                 :=   aOffset,
                hrPrev                  :=   hr);


hr := F_VN_GetNumberOfElements(ipContourList, nNumOfElem, hr);

IF nNumOfElem > 0 THEN
   FOR i:= 0 TO (nNumOfElem-1) BY 1 DO

      hr := F_VN_GetAt_ITcVnContainer(ipContourList, ipContour, i, hr);
```

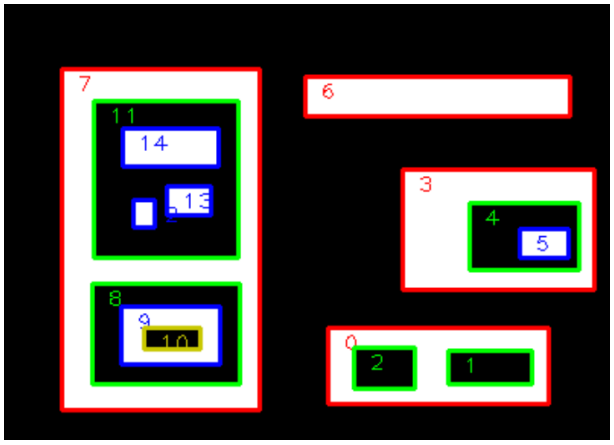
```
      hr := F_VN_GetAt_TcVnVector4_DINT(ipHierarchyList, aHierarchy, i, hr);

      // Count Parents and set aColor depending on parent number
      nParents := 0;
      aHierarchyTree := aHierarchy;
      WHILE SUCCEEDED(hr) AND_THEN aHierarchyTree[3] >= 0 DO
         nParents := nParents + 1;
         hr := F_VN_GetAt_TcVnVector4_DINT(ipHierarchyList, aHierarchyTree, aHierarchyTree[3], hr);
      END_WHILE

      // Draw contour and number into the result image
      sText := TO_STRING(i);
      hr := F_VN_UprightBoundingRectangle(ipContour, stRect, hr);
      hr := F_VN_PutText(sText, ipImageRes, stRect.nX +10, stRect.nY +15, TCVN_FT_HERSHEY_PLAIN, 1,
aColorList[(nParents) MOD 4], hr);
      hr := F_VN_DrawContours(ipContour, -1, ipImageRes, aColorList[(nParents) MOD 4], 2, hr);

      // Write Hierarchy
      hr := F_VN_PutText(sText, ipImageHierarchy, 10, TO_UDINT(10 + i*10), TCVN_FT_HERSHEY_PLAIN,
0.5, aColorList[(nParents) MOD 4], hr);
      FOR j := 0 TO 3 BY 1 DO
         sText := TO_STRING(aHierarchy[j]);
         hr := F_VN_PutText(sText, ipImageHierarchy, TO_UDINT(10 + (j+1)*22), TO_UDINT(10 + i*10),
TCVN_FT_HERSHEY_PLAIN, 0.5, aColorBlack, hr);
      END_FOR
   END_FOR
END_IF
```

Ergebnisse

Das Eingangsbild



Die gefundenen Konturen bei TCVN_CRM_TREE. Gleiche Farben geben Konturen auf gleicher Ebene wieder



Die Hierarchie zum obigen Ergebnisbild. Dabei bezieht sich jede Zeile jeweils auf eine Kontur und die Spaltenbedeutung ist wie folgt:

- Konturindex („farbige Spalte“)

- Index der nächsten Kontur auf gleicher Hierarchieebene
- Index der vorherigen Kontur auf gleicher Hierarchieebene
- Index der ersten Kind-Kontur
- Index der Eltern-Kontur

Die Zahl -1 gibt an, dass keine entsprechende Kontur existiert.

Ähnliche Beispiele

- Find Contour - Approximation Method [▶ 2875]
- Find Contour anstelle der Blob Detection [▶ 2873]

8.1.5.1.5 Template Matching mit Auswertung

In diesem Beispiel führen Sie ein Template Matching durch, um ein Bauteil auf einer Platine zu finden. Dazu verwenden Sie:

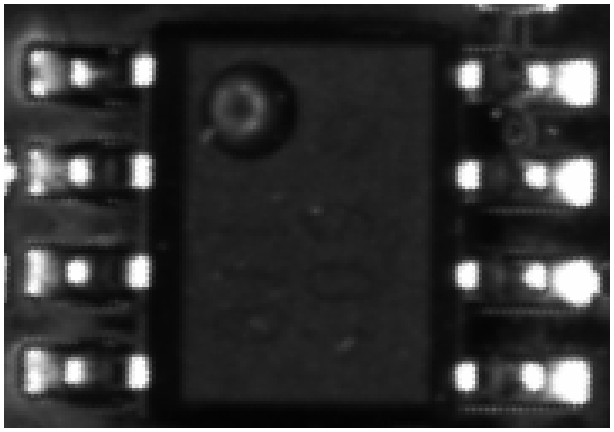
- F_VN_MatchTemplateAndEvaluate [▶ 1229]
- F_VN_DrawRectangle [▶ 1068]

Anwendung

Denken Sie daran, sowohl das Eingangsbild, als auch das Templatebild zu dem jeweiligen File Source hinzuzufügen. Über den Schwellwert `fMatchThreshold` können sie einschränken, wie gut ein Match sein muss, damit er akzeptiert wird.

Programm

In diesem Beispiel ist wichtig, dass Sie ein Templatebild zusätzlich zum normalen Eingangsbild benötigen, um nach Vorkommnissen des Templates auf dem Eingangsbild zu suchen.



Sofern beide Bilder zur Verfügung stehen, führen Sie das Template Matching automatisch mit der Funktion F_VN_MatchTemplateAndEvaluate [▶ 1229] durch.

```
hr := F_VN_MatchTemplateAndEvaluate(ipImageIn, ipImageTemplate, ipMatches, fMatchThreshold, hr);
```

Die gefundenen Matches, deren Übereinstimmungsgrad höher als der angegebene Schwellwert `fMatchThreshold` ist, befinden sich nun als Punkte in dem Container `ipMatches`. Jeder Punkt gibt die obere linkere Ecke jedes Vorkommnisses von `ipImageTemplate` in `ipImageIn` an.

Im Prinzip sind Sie nun mit dem Template Matching fertig. Beispielhaft greifen Sie im Folgenden auf die einzelnen Matches zu, um sie in das Bild zu malen. Dafür verwenden sie das Konstrukt zum Zugriff auf Container-Element, wie in der Sektion Container [▶ 163] erklärt.

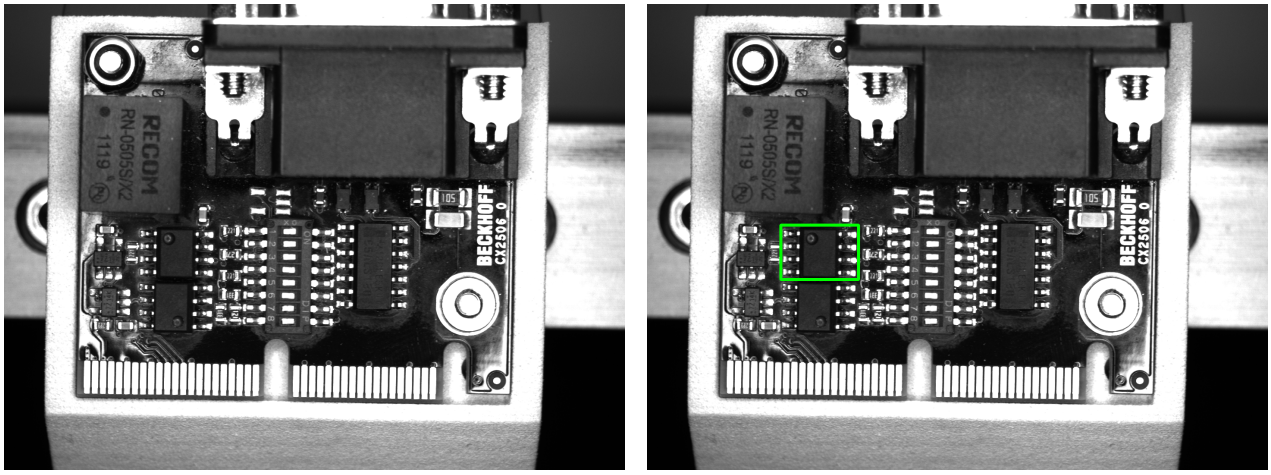
```
hr := F_VN_GetImageHeight(ipImageTemplate, nHeight, hr);
hr := F_VN_GetImageWidth(ipImageTemplate, nWidth, hr);
hr := F_VN_GetAt_TcVnPoint2_DINT(ipMatches, aPosition, 0, hr);
hr := F_VN_GetForwardIterator(ipMatches, ipIterator, hr);
IF SUCCEEDED(hr) AND ipIterator <> 0 THEN
    hr := ipIterator.TcQueryInterface(IID_ITcVnAccess_TcVnPoint2_DINT, ADR(ipAccess));
    IF SUCCEEDED(hr) AND ipAccess <> 0 THEN
        WHILE SUCCEEDED(hr) AND ipIterator.CheckIfEnd() <> S_OK DO
            hr := ipAccess.Get(aPosition);
            hr := F_VN_DrawRectangle(
                DINT_TO_UDINT(aPosition[0]),
                DINT_TO_UDINT(aPosition[1]),
                DINT_TO_UDINT(aPosition[0])+nWidth,
                DINT_TO_UDINT(aPosition[1])+nHeight,
                ipImageRes,
                aGreen,
                5,
                hr
            );
            hr := F_VN_IncrementIterator(ipIterator, hr);
        END_WHILE
    END_IF
END_IF
```

HINWEIS

Hohe Ausführungsdauer

Je nach Größe von Eingangs- und Templatebild kann die Ausführung des Template Matching im Vergleich zu anderen Funktionen sehr lange dauern. Achten Sie darauf, dass Sie die Zykluszeit entsprechend einstellen und ggfs. einen Watchdog verwenden!

Originalbild **Ergebnisbild**



8.1.5.2 Durchschnittsintensität in beliebigen Formen

In diesem Beispiel wird die durchschnittliche Intensität eines Bildes innerhalb eines manuell definierten Bildbereichs berechnet. Dazu wird eine kreisförmige Bildmaske [▶ 167] genutzt.

Dabei werden vor allem folgende Funktionen benutzt:

- F_VN_ImageAverageExp [▶ 1247]
- F_VN_DrawCircle [▶ 1031]

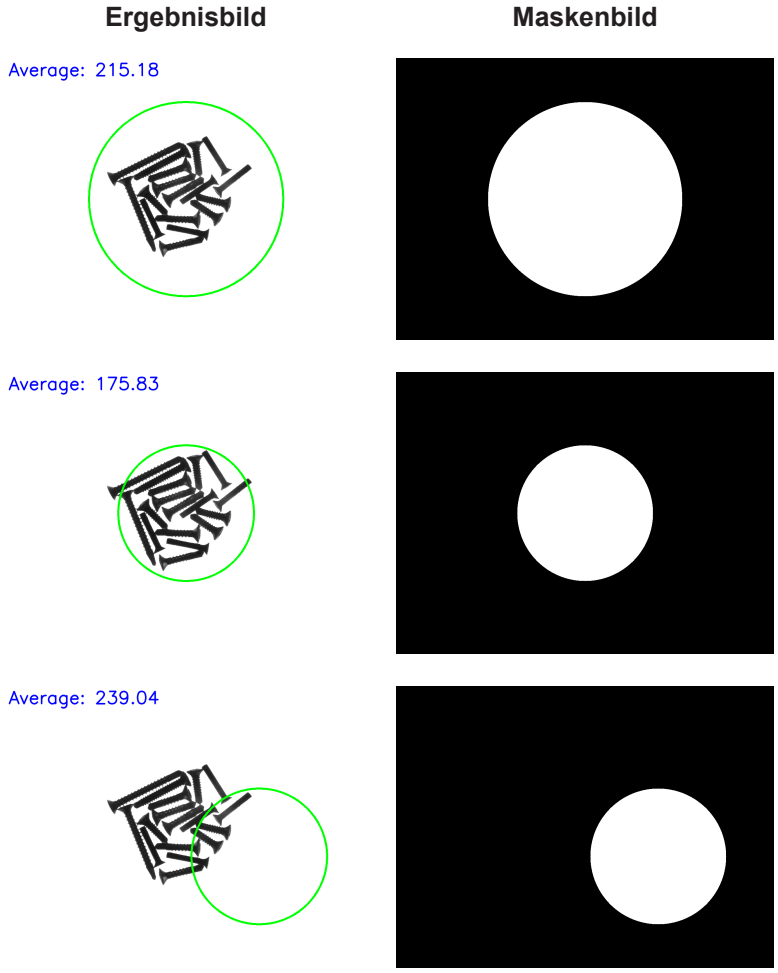
Erklärung

In dem Beispiel Statistische Bildmerkmale [▶ 2884] wird gezeigt, wie statistische Bildmerkmale (wie Durchschnitt, Maximum und Minimum) berechnet werden können. Häufig sollen diese Merkmale jedoch nicht für das gesamte Bild, sondern nur für bestimmte Bildbereiche berechnet werden. Sind diese Bereiche rechteckig, hilft das Setzen einer ROI [▶ 168]. Um die Berechnung auf beliebig geformte Bereiche zu beschränken, müssen Sie mit Bildmasken [▶ 167] arbeiten. Dies sind im Wesentlichen Bilder, deren Intensitätswerte beschreiben, welche Pixel in dem Originalbild berücksichtigt werden sollen. Somit kann eine Maske beliebige Formen annehmen; dieses Beispiel zeigt eine Maske in Kreisform.

Für dieses Beispiel wird ein quasi Binärbild mit einer Ansammlung von Schrauben betrachtet. Ein denkbarer Anwendungsfall wäre z. B., dass anhand des durchschnittlichen Intensitätswert annähernd die Menge der Schrauben bestimmt werden soll.

Anwendung

Durch Variieren der Parameter `nCircleShrink` und `aCenterOffset` können Sie die Größe und Position der kreisförmigen Maske verändern. Entsprechend dieser Maske verändert sich der durchschnittliche Intensitätswert: Wenn die Maske kleiner wird und weniger des weißen Hintergrunds zeigt, verringert sich der Durchschnittswert. Wenn die Maske sich verschiebt und mehr des weißen Hintergrunds zeigt, vergrößert sich der Durchschnittswert.



Bei Bedarf können auch andere Bilder in dieses Beispielprojekt eingeladen werden. Achten Sie darauf, dass dann ggfs. die Masken-Parameter angepasst werden müssen.

Code

Das Beispielbild wird über eine FileSource-Zustandsmaschine als `ipImageWork` eingeladen und mit folgendem Code verarbeitet.

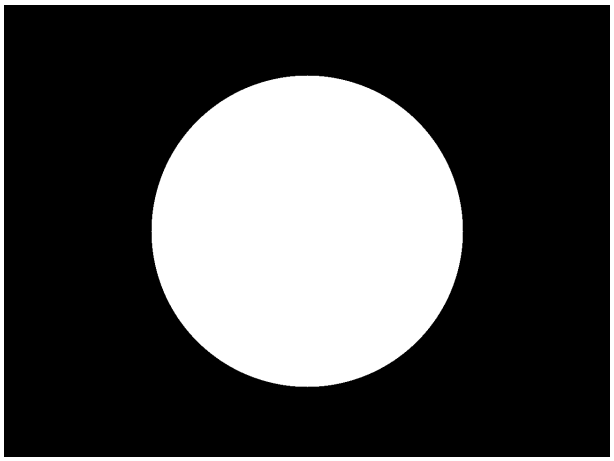
Zunächst werden die Eigenschaften der Maske festgelegt. Da die Maske ein Kreis ist, wird sie durch den Mittelpunkt `aCenter` und den Radius `nRadius` beschrieben. Als Mittelpunkt des Kreises wird der Bildmittelpunkt korrigiert durch das vom Nutzer einstellbares Offset `aCenterOffset` verwendet. Der Radius berechnet sich als Abstand vom Mittelpunkt zum nächstgelegenen Bildrand abzüglich des Werts `nCircleShrink`.

```
hr := F_VN_GetImageWidth(ipImageWork, nWidth, hr);
hr := F_VN_GetImageHeight(ipImageWork, nHeight, hr);
aCenter[0] := (nWidth / 2) + aCenterOffset[0];
aCenter[1] := (nHeight / 2) + aCenterOffset[1];
nRadius := (MIN(nWidth, nHeight) / 2) - nCircleShrink;
```

Als Maske wird ein neues Bild mit gleicher Größe wie das Eingangsbild `ipImageWork` erstellt und mit einem weißen Kreis auf schwarzem Hintergrund bemalt. Der Kreis erhält die errechneten Eigenschaften. Als Breite `nThickness` der Kreisumrandung wird der Wert `-1` gewählt, damit der Kreis ausgefüllt wird.

```
hr := F_VN_CreateImageAndSetPixels(ipImageMask, nWidth, nHeight, TCVN_ET_USINT, 1, aColorBlack, hr);
hr := F_VN_DrawCircle(
    nCenterX    :=  aCenter[0],
    nCenterY    :=  aCenter[1],
    nRadius     :=  nRadius,
    ipDestImage :=  ipImageMask,
    aColor      :=  aColorWhite,
    nThickness  :=  -1,
    hrPrev      :=  hr
);
```

Als Resultat steht nun ein Maskenbild `ipImageMask` mit der gewünschten Kreisform zur Verfügung:



Nun wird mit Hilfe dieser Maske `ipImageMask` der durchschnittliche Intensitätswert des Eingangsbildes `ipImageWork` im definierten kreisförmigen Bereich berechnet. Die Funktion F_VN_ImageAverageExp [▶ 1247] gibt das Array `aAverage` des Typs `TcVnVector4_LREAL` zurück, um die maximal möglich Anzahl an Bildkanälen abzudecken. Da das Eingangsbild in diesem Beispiel nur einen Kanal besitzt, ist das erste Array-Element der gewünschte Durchschnittswert und wird separat gespeichert.

```
hr := F_VN_ImageAverageExp(
    ipSrcImage  :=  ipImageWork,
    aAverage    :=  aAverage,
    ipMask      :=  ipImageMask,
    hrPrev      :=  hr
);
fAverageInMask := aAverage[0];
```

Schließlich werden der Maskenkreis sowie der errechnete Durchschnittswert in ein Ergebnisbild eingezeichnet.

```
hr := F_VN_ConvertColorSpace(ipImageWork, ipImageRes, TCVN_CST_GRAY_TO_RGB, hr);
hr := F_VN_DrawCircle(aCenter[0], aCenter[1], nRadius, ipImageRes, aColorGreen, 5, hr);
sText := CONCAT('Average: ', LREAL_TO_FMTSTR(fAverageInMask, 2, TRUE));
hr := F_VN_PutTextExp(sText, ipImageRes, 40, 60, TCVN_FT_HERSHEY_SIMPLEX, 2, aColorBlue, 3,
TCVN_LT_ANTIALIASED, FALSE, hr);
```

Für die Funktion `LREAL_TO_FMTSTR` wird die SPS-Bibliothek Tc2_Utilities benötigt.

8.1.5.3 Statistische Bildmerkmale

In diesem Beispiel werden statistische Merkmale (Min, Max, Median, Mittelwerkt, Standardabweichung) von Bildern berechnet. Dazu werden folgende Funktionen genutzt:

- F_VN_MinPixelValue [▶ 1258]
- F_VN_MaxPixelValue [▶ 1256]
- F_VN_ImageMedian [▶ 1253]
- F_VN_ImageAverageStdDev [▶ 1248]

Anwendung

Dieses Projekt enthält einen File Source, in den beliebige RGB-Bilder eingeladen werden können. Diese werden im Programm in Grauwertbilder konvertiert und hinsichtlich verschiedener Statistiken analysiert. Die Ergebniswerte werden ins Bild eingezeichnet und können im ADS Image Watch [▶ 141] unter `ipImageRes` betrachtet werden.

Programm

```
// Prepare images
hr := F_VN_ConvertColorSpace(ipImageIn, ipImageIn, TCVN_CST_RGB_TO_GRAY, hr);
hr := F_VN_CopyImage(ipImageIn, ipImageRes, hr);

// Calculate image statistics
hr := F_VN_MinPixelValue(
    ipImage     :=  ipImageIn,
    aMinValue   :=  aMin,
    aPosition   :=  aPos,
    hrPrev      :=  hr
);
hr := F_VN_MaxPixelValue(
    ipImage     :=  ipImageIn,
    aMaxValue   :=  aMax,
    aPosition   :=  aPos,
    hrPrev      :=  hr
);
hr := F_VN_ImageMedian(
    ipSrcImage  :=  ipImageIn,
    aMedian     :=  aMedian,
    hrPrev      :=  hr
);
hr := F_VN_ImageAverageStdDev(
    ipSrcImage  :=  ipImageIn,
    aAverage    :=  aAverage,
    aStdDev     :=  aStdDev,
    hrPrev      :=  hr
);

// Draw results
sText := CONCAT('Min: ', LREAL_TO_FMTSTR(aMin[0], 2, TRUE ));
hr := F_VN_PutLabel(sText, ipImageRes, nTextX, (nTextYBase + (1 * nTextYIncrement)), fFontScale,
hr);
sText := CONCAT('Max: ', LREAL_TO_FMTSTR(aMax[0], 2, TRUE ));
hr := F_VN_PutLabel(sText, ipImageRes, nTextX, (nTextYBase + (2 * nTextYIncrement)), fFontScale,
hr);
sText := CONCAT('Median: ', LREAL_TO_FMTSTR(aMedian[0], 2, TRUE ));
hr := F_VN_PutLabel(sText, ipImageRes, nTextX, (nTextYBase + (3 * nTextYIncrement)), fFontScale,
hr);
sText := CONCAT('Average: ', LREAL_TO_FMTSTR(aAverage[0], 2, TRUE ));
hr := F_VN_PutLabel(sText, ipImageRes, nTextX, (nTextYBase + (4 * nTextYIncrement)), fFontScale,
hr);
sText := CONCAT('StdDev: ', LREAL_TO_FMTSTR(aStdDev[0], 2, TRUE ));
hr := F_VN_PutLabel(sText, ipImageRes, nTextX, (nTextYBase + (5 * nTextYIncrement)), fFontScale,
hr);

hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, hr);
hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, hr);
```

Für die Funktion `LREAL_TO_FMTSTR` wird die SPS-Bibliothek Tc2_Utilities benötigt.

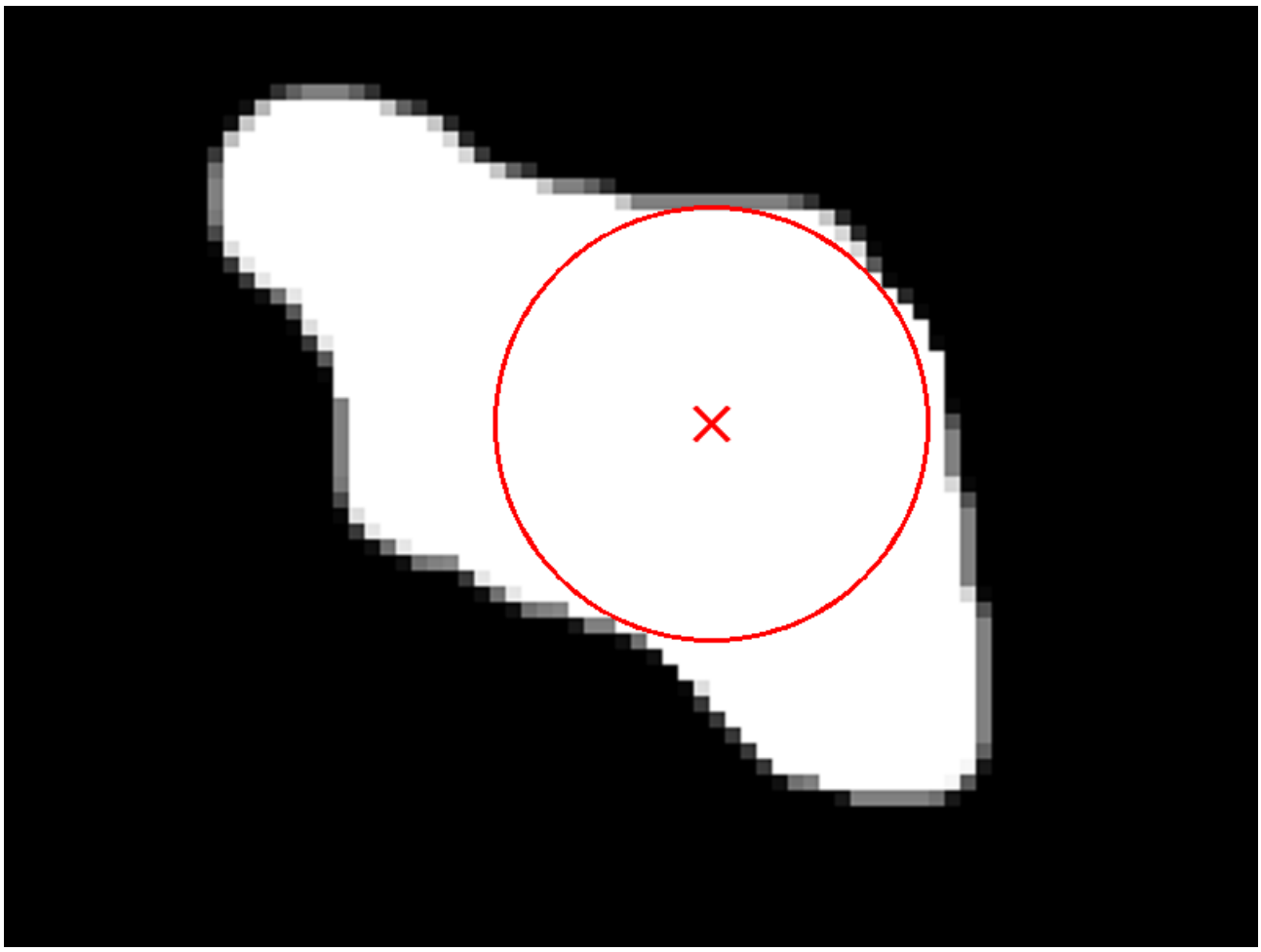
8.1.5.4 Innerer Kreis mittels Distanztransformation

In diesem Beispiel finden Sie den inneren Kreis eines unförmigen Objekts mittels der Distanztransformation. Dazu verwenden Sie im Wesentlichen folgende Funktionen:

- F_VN_DistanceTransformation [▶ 1242]
- F_VN_DrawCircle [▶ 1031]

Erklärung

Äußere Konturapproximationen wie ein umschließendes Rechteck oder die konvexe Hülle können über entsprechende API-Funktionen berechnet werden. Das Finden eines inneren Kreises mit maximaler Größe innerhalb einer Kontur kann hingegen mittels einer Distanztransformation gelöst werden.



Dafür wird mit einer Distanztransformation der Punkt ermittelt, der den größten euklidischen Abstand zu den Objekträndern hat. Dieser Punkt ist dann der Mittelpunkt des inneren Kreises des Objekts.

Wenn Sie dieses Beispiel mit eigenen Bildern benutzen, sollten Sie den Schwellwert und die Binarisierung über die Variablen `bInvert` und `fThreshold` anpassen. Mittels den Parametern `eDistanceType` und `eMaskSize` können Sie die Parameter der Funktion F_VN_DistanceTransformation [▶ 1242] gezielt verändern.

Programm

Der erste Schritt ist, das Eingangsbild zu binarisieren. Dadurch wird eindeutig zwischen Objekt und Hintergrund unterschieden. Die Binarisierung ist für die Distanztransformation wichtig, da die Distanztransformation nur auf Pixel mit dem Wert 0 reagiert. Falls das Objekt schwarz auf weißem Hintergrund ist, muss mittels des Schwellwerttyps `BINARY_INV` invertiert werden.

```
IF bInvert THEN
    eThresholdType := TCVN_TT_BINARY_INV;
ELSE
    eThresholdType := TCVN_TT_BINARY;
END_IF
hr := F_VN_Threshold(ipImageIn, ipImageWork, fThreshold, 255, eThresholdType, hr);
```

Durch die Distanztransformation erlangen Sie ein Bild, in dem jeder Pixelwert den Abstand zum nächsten 0-wertigen Pixel anzeigt. Das Pixel mit dem höchsten Wert ist somit das Pixel mit dem größten Abstand. Wenn Sie als Distanztyp `eDistanceType` die L2-Norm (euklidisch Abstand) verwenden, können Sie dieses Pixel als Mittelpunkt eines Kreises interpretieren, der das Objekt so viel wie möglich ausfüllt. Daher ermitteln Sie das höchstwertigsten Pixel mit der Funktion F_VN_MaxPixelValue [▶ 1256] und verwenden dessen Position als Kreismittelpunkt sowie dessen Wert als Kreisradius.

```
hr := F_VN_DistanceTransformation(
    ipSrcImage      :=  ipImageWork,
    ipDestImage     :=  ipImageWork,
    eDistanceType   :=  TCVN_DT_L2,
    eMaskSize       :=  TCVN_DTM_5,
```

```
    hrPrev          :=  hr
);
hr := F_VN_MaxPixelValue(ipImageWork, aMax, aPos, hr);
fRadius := aMax[0];
aCenter := aPos;
```

Die Maskengröße `eMaskSize` bestimmt die Genauigkeit der Abstandsberechnung und hat wenig Einfluss auf das Ergebnis.

Schließlich malen Sie den Kreis anhand des ermittelten Mittelpunkts und Radius in ein Ergebnisbild ein, um die Ausführung validieren zu können.

```
hr := F_VN_DrawCircle(
    nCenterX    :=  TO_UDINT(aCenter[0]),
    nCenterY    :=  TO_UDINT(aCenter[1]),
    nRadius     :=  TO_UDINT(fRadius),
    ipDestImage :=  ipImageRes,
    aColor      :=  aRed,
    nThickness  :=  2,
    hrPrev      :=  hr
);
hr := F_VN_DrawPointExp(TO_UDINT(aCenter[0]), TO_UDINT(aCenter[1]), ipImageRes, TCVN_DS_X, aRed, 10,
2, TCVN_LT_ANTIALIASED, hr);
```

8.1.6 Image Color and Contrast Processing

8.1.6.1 Color Similarity mit RGB-Referenz-Farbe

In diesem Beispiel wird über die Funktion F_VN_ReferenceColorSimilarityExp_TcVnVector3_LREAL [▶ 1288] die Ähnlichkeit jedes Pixels zu 3 Referenzfarben berechnet und jeweils als Grauwertbild zurückgegeben. Anhand dieser Grauwertbilder werden anschließend die Objekte in den Referenzfarben detektiert.

Variablen

```
hr                  :   HRESULT;

// Images
ipImageIn           :   ITcVnImage;
ipImageInDisp       :   ITcVnDisplayableImage;

ipImageThres        :   ITcVnImage;
ipImageThresDisp    :   ARRAY [0..2] OF ITcVnDisplayableImage;
ipImageWorkCol      :   ITcVnImage;
ipImageWorkColDisp  :   ARRAY [0..2] OF ITcVnDisplayableImage;

ipImageRes          :   ITcVnImage;
ipImageResDisp      :   ITcVnDisplayableImage;

// Color
iColor              :   INT;
aColorTxt           :   ARRAY[0..2] OF STRING := [ 'RED', 'GREEN', 'BLUE' ];
aColor              :   ARRAY[0..2] OF TcVnVector4_LREAL := [ [150, 0, 0], [0, 255, 0],
[0, 0, 255] ];
aColorRef           :   ARRAY[0..2] OF TcVnVector3_LREAL := [ [255, 75, 60], [40, 140, 95],
[40, 140, 190] ];

// Contours
ipContourList       :   ITcVnContainer;
ipIterator          :   ITcVnForwardIterator;
aOffset             :   TcVnPoint;
ipContour           :   ITcVnContainer;
fArea               :   LREAL;
aCenter             :   TcVnPoint2_LREAL;
```

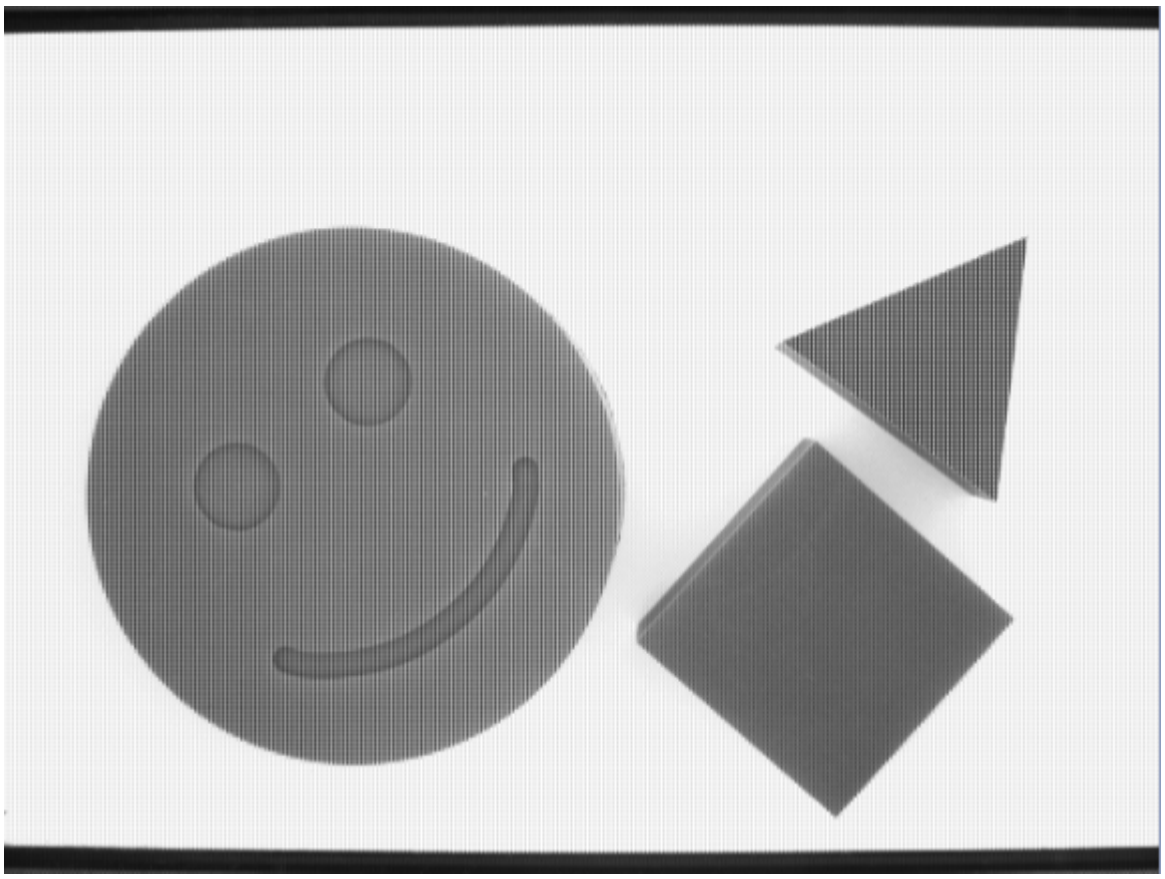
Code

```
// Attention: With other images another color space transformation could be necessary
hr := F_VN_ConvertColorSpace(ipImageIn, ipImageRes, TCVN_CST_Bayer_RG_TO_RGB, hr);

FOR iColor := 0 TO 2 DO

   // Compute the Color Similarity to a Reference Color
   hr := F_VN_ReferenceColorSimilarityExp_TcVnVector3_LREAL(
                  ipSrcImage        := ipImageRes,
```

```
                  ipDestImage       := ipImageWorkCol,
                  aRefColor         := aColorRef[iColor],
                  fVariance         := 0.1,
                  fLuminanceWeight  := 0.2,
                  hrPrev            := hr);

   hr := F_VN_Threshold(ipImageWorkCol, ipImageThres, 200, 255, TCVN_TT_Binary, hr);

   // Find all objects / contours in the black and white image
   hr := F_VN_FindContours(ipImageThres, ipContourList, hr);
   hr := F_VN_GetForwardIterator(ipContourList, ipIterator, hr);

   // Filter the objects by size and draw the contours
   WHILE SUCCEEDED(hr) AND_THEN ipIterator.CheckIfEnd() <> S_OK DO
      hr := F_VN_GetContainer(ipIterator, ipContour, hr);
      hr := F_VN_IncrementIterator(ipIterator, hr);

      // Filter contours by size
      hr := F_VN_ContourArea(ipContour, fArea, hr);
      IF fArea > 5000 THEN
         // Draw Results into an Image
         hr := F_VN_DrawContours(ipContour, -1, ipImageRes, aColor[iColor], 3, hr);
         hr := F_VN_ContourCenterOfMass(ipContour, aCenter, hr);
         hr := F_VN_PutText(aColorTxt[iColor], ipImageRes, LREAL_TO_UDINT(aCenter[0])-30,
LREAL_TO_UDINT(aCenter[1])+10, TCVN_FT_HERSHEY_PLAIN, 2, aColor[iColor],hr);
      END_IF
   END_WHILE

   // Display effect of the Color-Similarity-Function
   hr := F_VN_TransformIntoDisplayableImage(ipImageThres, ipImageThresDisp[iColor], hr);
   hr := F_VN_TransformIntoDisplayableImage(ipImageWorkCol, ipImageWorkColDisp[iColor], hr);
END_FOR
```

Ergebnis

Das Eingangsbild im Bayer RG Format

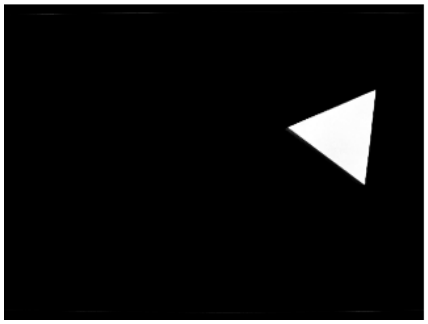


Nach der Konvertierung des Eingangsbildes mit der F_VN_ConvertColorSpace [▶ 1270] Funktion in ein RGB Bild, kann mit der F_VN_ReferenceColorSimilarityExp_TcVnVector3_LREAL [▶ 1288] die Farbähnlichkeit jedes Pixels zur Referenzfarbe ermittelt werden. Im zurück gegebenen Ergebnisbild entspricht:

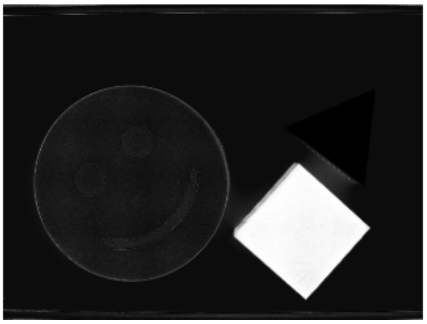
- der Pixelwert 255 (weiß) einer 100%igen Übereinstimmung mit der Referenzfarbe
- der Pixelwert 0 (schwarz) einer 0%igen Übereinstimmung mit der Referenzfarbe
- und die Pixelwerte zwischen 0 und 255 einer teilweisen Übereinstimmung mit der Referenzfarbe.

```
// Compute the Color Similarity to a Reference Color
hr := F_VN_ReferenceColorSimilarityExp_TcVnVector3_LREAL(
                ipSrcImage        := ipImageRes,
                ipDestImage       := ipImageWorkCol,
                aRefColor         := aColorRef[iColor],
                fVariance         := 0.1,
                fLuminanceWeight  := 0.2,
                hrPrev            := hr);
```

aColorRef := [255,75,60]; (RED)



aColorRef := [40,140,95]; (GREEN)

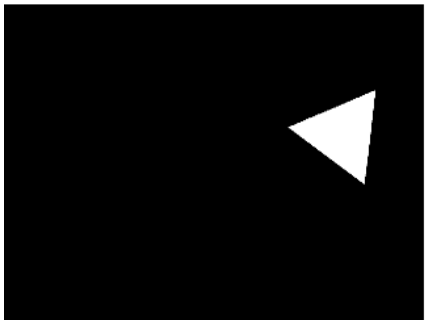


aColorRef := [40,140,190]; (BLUE)

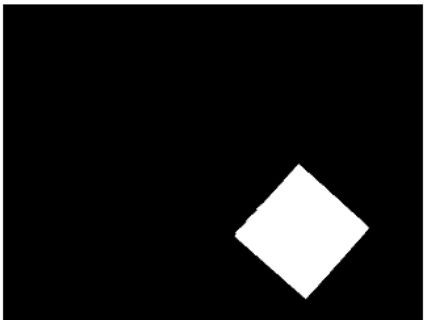
Über einen Schwellwert (F_VN_Threshold [▶ 1363]) wird anschließend definiert, wie stark die Übereinstimmung zur Referenzfarbe sein soll, um ihr zugeordnet zu werden.

```
hr := F_VN_Threshold(ipImageWorkCol, ipImageThres, 200, 255, TCVN_TT_Binary, hr);
```

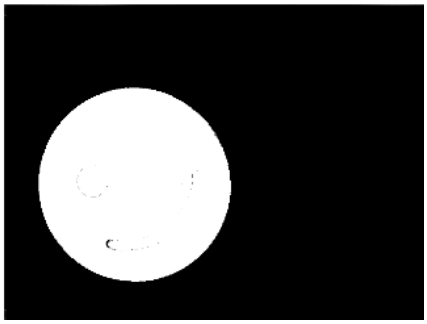
aColorRef := [255,75,60]; (RED)



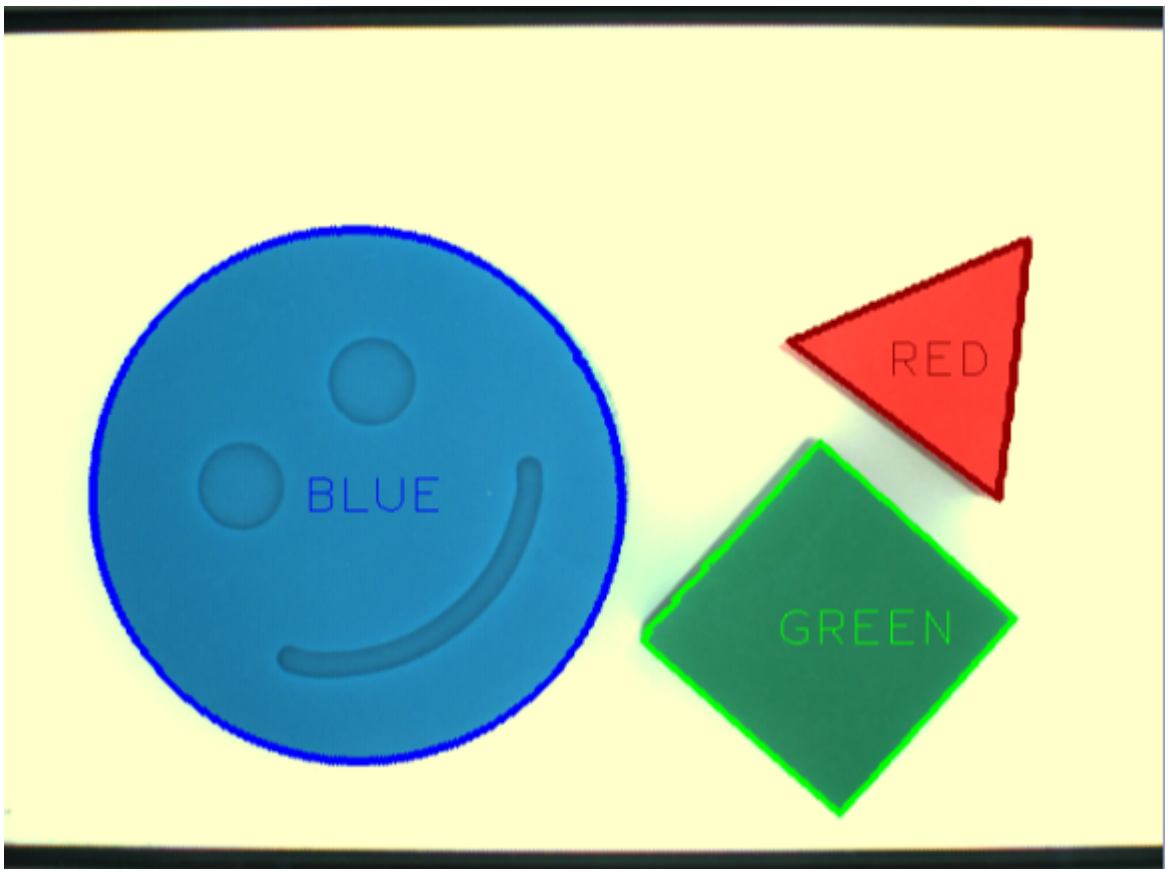
aColorRef := [40,140,95]; (GREEN)



aColorRef := [40,140,190]; (BLUE)



Das Ergebnisbild, nach anschließender Detektion.



Ähnliches Beispiel

- Check Color Range mit RGB-Range [▶ 2897]

8.1.6.2 Darstellung mit Farbtabellen

In diesem Beispiel wenden Sie Farbtabellen auf Grauwertbilder an. Sie nutzen im Wesentlichen die folgenden Funktionen:

- F_VN_GenerateColorMap [▶ 1270]
- F_VN_CreateContainerFromArray [▶ 776]
- F_VN_GenerateCustomColorMap [▶ 1272]
- F_VN_ApplyColorMap [▶ 1265]

Erklärung

Es gibt mehrere Gründe, warum Sie eine Farbtabelle auf ein monochromes Bild anwenden wollen. Dazu zählen z. B. eine angenehmere Optik und eine für das menschliche Auge klarere Darstellung des Wertebereichs. Dies ist vor allem für 16-Bit Bilder relevant, da das Wertespektrum wesentlich besser erkennbar ist, wenn es auf drei Farbkanäle verteilt wird.

Je nach Anwendungsfall möchten Sie verschiedene Farbtabellen verwenden. TwinCAT Vision stellt über das Enum ETcVnColorMap [▶ 181] einige vordefinierte Farbverläufe zur Verfügung. Ebenso können Sie eigene Farbverläufe definieren und daraus Farbtabellen erstellen.

In diesem Beispiel wenden Sie verschiedene Farbtabellen auf den folgenden Grauwertverlauf (mit einem Wertebereich von 16 Bit) an.

Abb. 26: Grauwertverlauf

Selbstverständlich können sie auch eigene Bilder verwenden. Achten Sie darauf, dass Sie nur 1-kanlige Bilder verwenden können bzw. vorher eine entsprechende Konvertierung mittels F_VN_ConvertColorSpace [▶ 1270] durchführen müssen.

Programm

Bevor eine Farbtabelle auf ein Bild angewandt werden kann, muss sie erst erstellt werden. Dazu muss bekannt sein, ob die Farbtabelle für 8-Bit oder 16-Bit Bilder sein soll, da sie abhängig davon verschieden viele Einträge benötigt (siehe Farbtabellen). Dementsprechend müssen Sie den Parameter `eColorMapSize` vom Typ ETcVnColorMapSize [▶ 186] bei 8 Bit auf `256` und bei16 Bit auf `65536` passend zur Bittiefe des Eingangsbildes `ipImageIn` setzen. Sie erreichen dies durch eine Abfrage der Elementgröße des Pixelformats mittels F_VN_GetImageInfo [▶ 820]:

```
hr := F_VN_GetImageInfo(ipImageIn, stImageInfo, hr);
IF stImageInfo.stPixelFormat.nElementSize = 8 THEN
    eColorMapSize := TCVN_CMS_256;
ELSIF stImageInfo.stPixelFormat.nElementSize = 16 THEN
    eColorMapSize := TCVN_CMS_65536;
END_IF
```

Das Generieren einer Farbtabelle erfolgt anhand eines Farbverlaufs, den Sie entweder manuell definieren oder aus einer Vorlage verwenden. Dieses Beispiel nimmt die Unterscheidung anhand des Parameters `bUseCustomColors` vor.

```
IF bUseCustomColors THEN
    // create color map from custom-defined colors
    <...>
ELSE
    // create color map from preset
    <...>
END_IF
```

Sie definieren einen Farbverlauf manuell, indem Sie einen Container mit mehreren Einträgen vom Typ TcVnVector3_REAL erstellen. Jeder Eintrag stellt eine Farbe dar. Die Farben werden beim Erstellen der Farbtabelle als gleichverteilt über den Definitionsbereich interpretiert; alle übrigen Werte werden für jeden der drei Kanäle linear interpoliert. Hier wird ein Farbverlauf aus dem Array `aColors` mit den drei Farben Rot, Grün und Blau verwendet. Sie erstellen die Farbtabelle dann mit der Funktion F_VN_GenerateCustomColorMap.

```
VAR
    aColors: ARRAY [0..(N_COLORS-1)] OF TcVnVector3_REAL := [
        [255, 0, 0],
        [0, 255, 0],
        [0, 0, 255]
    ];
END_VAR

hr := F_VN_CreateContainerFromArray(ADR(aColors), ipColors, ContainerType_Vector_TcVnVector3_REAL,
N_COLORS, hr);
hr := F_VN_GenerateCustomColorMap(
    ipColorMap      :=  ipColorMap,
    ipInitialColors :=  ipColors,
    eColorMapSize   :=  eColorMapSize,
    hrPrev          :=  hr
);
```

Um einen vordefinierten Farbverlauf zu verwenden, nutzen Sie die Funktion F_VN_GenerateColorMap [▶ 1270] und setzen den Parameter `eColorMap` auf den gewünschten Farbverlauf aus dem Enum ETcVnColorMap [▶ 181]. Hier ist dieser initial auf `JET` gesetzt.

```
VAR
    eColorMap: ETcVnColorMap := TCVN_CM_JET;
END_VAR

hr := F_VN_GenerateColorMap(
    ipColorMap      :=  ipColorMap,
    eColorMap       :=  eColorMap,
    eColorMapSize   :=  eColorMapSize,
    hrPrev          :=  hr
);
```

Die erstellte Farbtabelle ist ein Container [▶ 163], der für jeden möglichen Wert im Eingangsbild eine zu vergebende Farbe gespeichert hat. Daher können Sie eine Farbtabelle auch selbst erstellen, indem Sie alle 256 bzw. 65536 Einträge festlegen und daraus direkt einen Container erstellen. In den meisten Fällen ist die Verwendung der API-Funktionen jedoch einfacher.

Schließlich wenden Sie die erstellte Farbtabelle auf das Eingangsbild an und erhalten das gewünschte Ergebnis. Das Ergebnisbild immer ein 3-kanaliges 8-Bit Bild.

```
hr := F_VN_ApplyColorMap(
    ipSrcImage      :=  ipImage,
    ipDestImage     :=  ipImage,
    ipColorMap      :=  ipColorMap,
    hrPrev          :=  hr
);
```

Die Anwendung der vordefinierten JET-Farbtabelle auf das oben gezeigte Eingangsbild erzeugt folgendes Ergebnis:

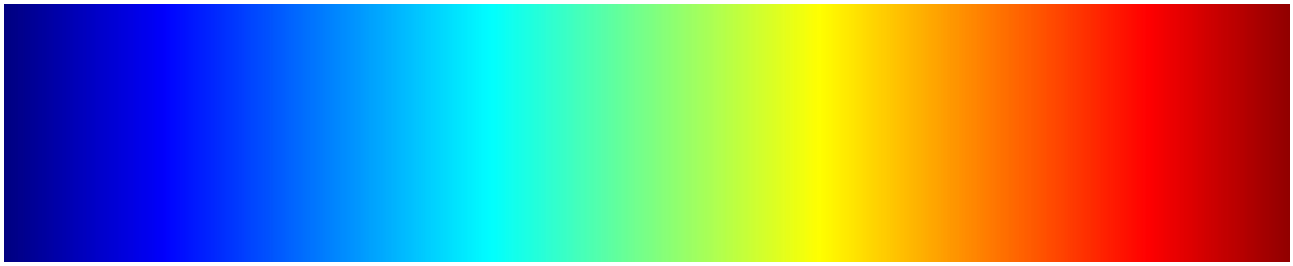


Abb. 27: Farbverlauf als Ergebnis nach Anwendung der JET-Farbtabelle

8.1.7 Image Filtering

8.1.7.1 Weichzeichnungsfilter

In diesem Beispiel werden folgende Weichzeichnungsfilter verglichen:

- Bilateral-Filter [▶ 1305]
- Gauß-Filter [▶ 1324]
- Median-Filter [▶ 1335]
- Mean-Filter umgesetzt als Custom-Filter [▶ 1316]

Erläuterung

In manchen Fällen soll ein Bild geglättet bzw. von Rauschen befreit werden. Bei Aufgaben wie Code Reading ist es gleichzeitig nötig, einen guten Kontrast und gute Kanten auf dem Bild zu erhalten, damit die nachfolgenden Algorithmen erfolgreich darauf arbeiten können.

Für die Kombination von Bildglättung bei gleichzeitigem Erhalt von Kanten eignet sich der **Bilateral-Filter** besonders gut. Er berechnet für jede Pixelintensität einen gewichteten Durchschnittswert der umliegenden Pixel, wobei die Gewichtung nicht nur von der Entfernung eines Pixels, sondern ebenfalls von dem Intensitätsunterschied abhängt. Dadurch bleiben scharfe Kanten erhalten.

Der **Gauß-Filter** glättet Bilder mit Hilfe einer zweidimensionalen diskreten Annäherung an die Gauß'sche Glockenfunktion. Dadurch kann das Bildrauschen vermindert werden, wobei kleinere Strukturen verloren gehen, gröbere Strukturen bleiben dagegen erhalten. Kanten verwischen daher weniger als bei den beiden nachfolgenden Filtern, größere Störungen werden jedoch nicht so gut entfernt.

Der **Median-Filter** sortiert alle Pixelwerte der vorgegebenen Maske aufsteigend und wählt dann den Median (den mittleren) Wert aus, auf den das entsprechende Pixel dann gesetzt wird. Dieser Filter ist gut geeignet, um Strukturen (Störungen) zu unterdrücken, die kleiner als die angewendete Filtermaske sind und verhält sich sehr robust gegenüber Ausreißern. Dadurch werden z. B. die kleinen Salz- und Pfeffer-Störungen am besten entfernt, was aber den Nachteil hat, dass dadurch auch die anderen Detailinformationen verloren gehen.

Zur Realisierung einer Mittelwertbildung (Mean) wurde die **Custom-Filter** Funktion verwendet. Dabei wird der Wert jedes Pixels durch den Mittelwert der Kernel-Pixel ersetzt, wodurch ein Weichzeichnen bewirkt wird. Dadurch werden Störungen (Rauschen) unterdrückt, Details und Kanten verschmieren aber allmählich mit steigender Kernelanzahl. Da die Custom Filter-Funktion viele verschiedene Möglichkeiten bietet eigene Filter anzuwenden, wird zur Standardisierung des Filter-Kernels noch die `F_VN_InitMatrixStruct` Funktion benötigt.

Variablen

```
// Bilateral Filter
ipImageBilateral         :   ITcVnImage;
ipImageBilateralDisp     :   ITcVnDisplayableImage;
nBilateral_Diameter      :   DINT  := 7;
fBilateral_SigmaColor    :   LREAL := 100;
fBilateral_SigmaSpace    :   LREAL := 100;

// Gaussian Filter
ipImageGaussian          :   ITcVnImage;
ipImageGaussianDisp      :   ITcVnDisplayableImage;
nFilterWidth             :   UDINT := 7;
nFilterHeight            :   UDINT := 7;

// Median Filter
ipImageMedian            :   ITcVnImage;
ipImageMedianDisp        :   ITcVnDisplayableImage;
nMedian_FilterSize       :   UDINT := 7;

// Custom Filter e.g. Mean
ipImageCustom            :   ITcVnImage;
ipImageCustomDisp        :   ITcVnDisplayableImage;
stKernelMatrix           :   TcVnMatrix;
// 7x7 Mean Filter Kernel with weights of 1/49 ~ 0.0204081632653
aMatrixArray7x7          :   ARRAY [0..6,0..6] OF REAL := [49(0.0204081632653)];
```

Code

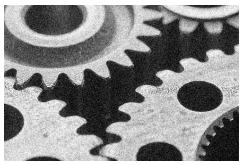
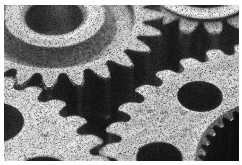
```
hr := F_VN_BilateralFilter(
        ipSrcImage      := ipImageIn,
        ipDestImage     := ipImageBilateral,
        nDiameter       := nBilateral_Diameter,
        fSigmaColor     := fBilateral_SigmaColor,
        fSigmaSpace     := fBilateral_SigmaSpace,
        hrPrev          := hr);

hr := F_VN_GaussianFilter(
        ipSrcImage      := ipImageIn,
        ipDestImage     := ipImageGaussian,
        nFilterWidth    := nFilterWidth,
        nFilterHeight   := nFilterHeight,
        hrPrev          := hr);

hr := F_VN_MedianFilter(
        ipSrcImage      := ipImageIn,
        ipDestImage     := ipImageMedian,
        nFilterSize     := nMedian_FilterSize,
        hrPrev          := hr);

// Mean-Filter with Custom Filter Function
hr := F_VN_InitMatrixStruct(
        pSrcBuffer      := ADR(aMatrixArray7x7),
        stDestMatrix    := stKernelMatrix,
        nRows           := 7,
        nCols           := 7,
        eElementType    := TCVN_ET_REAL,
        hrPrev          := hr);

hr := F_VN_CustomFilter(
        ipSrcImage      := ipImageIn,
        ipDestImage     := ipImageCustom,
```

```
        eDestDepth      := TCVN_ET_USINT,
        stKernel        := stKernelMatrix,
        hrPrev          := hr);
```

Ergebnis

Das unbearbeitete Originalbild (1. Zeile) weist im Detail bereits feine Strukturen auf den Oberflächen der Zahnräder auf. Um die Filterwirkungen zu verdeutlichen, wurden zusätzliche Störungen durch ein Gauss- (2. Zeile) und ein Salz- und Pfeffer-Rauschen (3. Zeile) zu dem Originalbild hinzugefügt.

Damit die Wirkung der verschiedenen Filter möglichst vergleichbar ist, wird bei allen eine Kernelgröße von 7 x 7 verwendet. Durch andere Parameterwerte oder Kernelgrößen, sowie weitere Einstellmöglichkeiten bei den EXP-Varianten der Funktionen lassen sich im Einzelnen auch noch andere Wirkungen erzielen. Da die Filterwirkung stark vom Inhalt des Eingangsbildes abhängt, empfiehlt es sich, mehrere Filter mit verschiedenen Parametereinstellungen zu testen, um dann direkt zu vergleichen und den besten Filter für den Anwendungsfall zu finden. In der folgenden Tabelle erhalten Sie eine Übersicht über die Ergebnisbilder:

Originalbilder	Bilateraler Filter	Gauss-Filter	Median-Filter	Mean-Filter
				
				

HINWEIS

Hohe oder schwankende Ausführungszeiten

Die Dauer der Ausführung eines Filters hängt im Wesentlichen vom Typ des Filters, sowie der Bild- und Kernelgröße ab. Beachten Sie daher, dass eine Filteroperation wesentlich mehr Zeit in Anspruch nehmen kann als andere Funktionen, passen Sie daher die Zykluszeit der Task entsprechend an.

Wenn die Filterparameter während der Laufzeit verändert werden, insbesondere die Kernelgröße, ist es empfehlenswert Watchdogs [▶ 156] zu verwenden. Beim Median Filter kann die Laufzeit prinzipbedingt immer etwas schwanken.

Weiterhin können Job-Tasks [▶ 66] zu einer Reduzierung der Gesamtrechenzeit führen, was in Abhängigkeit des Systems zu testen ist.

8.1.7.2 Kantendetektionsfilter

In diesem Beispiel werden folgende Kantendetektionsfilter verglichen:

- Sobel-Filter [▶ 1350]
- Scharr-Filter [▶ 1342]
- Laplace-Filter [▶ 1328]

Erläuterung

Der Sobel-Filter ist ein einfacher linearer Filter, um Gradienten in einem Bild zu markieren. Der Scharr-Filter ist dazu sehr ähnlich; lediglich die Ausprägung des Filter-Kernels ist etwas anders. Der Scharr-Filter hat eine bessere Isotropie, d. h. die Detektion von Kanten ist richtungsunabhängig.

Der Laplace-Filter detektiert Kanten basierend auf den Nullstellen der diskreten 2. Ableitung des Bildes. Dieses Verfahren ist vergleichsweise rauschanfällig.

Variablen

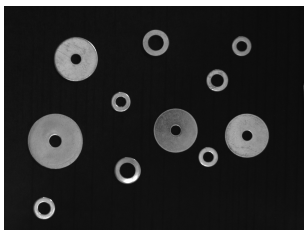
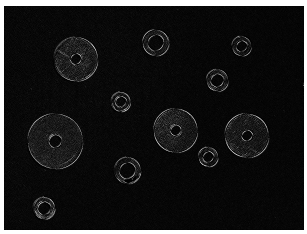
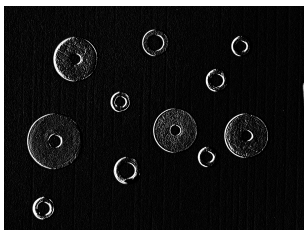
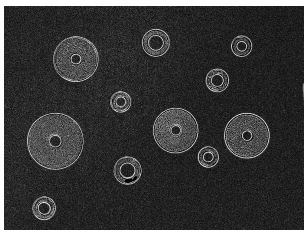
```
// Sobel
ipImageSobel         :   ITcVnImage;
ipImageSobelDisp     :   ITcVnDisplayableImage;
eSobel_DestDepth     :   ETcVnElementType := TCVN_ET_USINT;
aSobel_DerivOrder    :   ARRAY [0..1] OF UDINT := [1, 1];
nSobel_KernelSize    :   UDINT := 3;
fSobel_Scale         :   LREAL := 10;
fSobel_Delta         :   LREAL := 0;
eSobel_BorderExtra   :   ETcVnBorderInterpolationMethod := TCVN_BIM_DEFAULT;

// Scharr
ipImageScharr        :   ITcVnImage;
ipImageScharrDisp    :   ITcVnDisplayableImage;
eScharr_DestDepth    :   ETcVnElementType := TCVN_ET_USINT;
eScharr_FilterDir    :   ETcVnFilterDirection := TCVN_FD_X;
fScharr_Scale        :   LREAL := 1;
fScharr_Delta        :   LREAL := 0;
eScharr_BorderExtra  :   ETcVnBorderInterpolationMethod := TCVN_BIM_DEFAULT;

// Laplace
ipImageLaplace       :   ITcVnImage;
ipImageLaplaceDisp   :   ITcVnDisplayableImage;
eLaplace_DestDepth   :   ETcVnElementType := TCVN_ET_USINT;
nLaplace_KernelSize  :   UDINT := 3;
fLaplace_Scale       :   LREAL := 10;
fLaplace_Delta       :   LREAL := 0;
eLaplace_BorderExtra :   ETcVnBorderInterpolationMethod := TCVN_BIM_DEFAULT;
```

Code

```
// Execute the Sobel Filter
hr := F_VN_SobelFilterExp(
         ipSrcImage   := ipImageIn
         ipDestImage  := ipImageSobel,
         eDestDepth   := eSobel_DestDepth,
         nXOrder      := aSobel_DerivOrder[0],
         nYOrder      := aSobel_DerivOrder[1],
         nKernelSize  := nSobel_KernelSize,
         fScale       := fSobel_Scale,
         fDelta       := fSobel_Delta,
         eBorderType  := eSobel_BorderExtra,
         hrPrev       := hr);

// Execute the Scharr Filter
hr := F_VN_ScharrFilterExp(
         ipSrcImage      := ipImageIn,
         ipDestImage     := ipImageScharr,
         eDestDepth      := eScharr_DestDepth,
         eFilterDirection:= eScharr_FilterDir,
         fScale          := fScharr_Scale,
         fDelta          := fScharr_Delta,
         eBorderType     := eScharr_BorderExtra,
         hrPrev          := hr);

// Execute the Laplacian Filter
hr := F_VN_LaplacianFilterExp(
         ipSrcImage   := ipImageIn,
         ipDestImage  := ipImageLaplace,
         eDestDepth   := eLaplace_DestDepth,
         nKernelSize  := nLaplace_KernelSize,
         fScale       := fLaplace_Scale,
         fDelta       := fLaplace_Delta,
         eBorderType  := eLaplace_BorderExtra,
         hrPrev       := hr);
```

Ergebnisse

Das Beispielprojekt enthält die Funktionsaufrufe der drei Filter. Die Parameter-Konfiguration und deren Auswirkung auf ein Beispielbild können in diesem Beispiel ausprobiert werden.

Originalbild	Sobel-Filter	Scharr-Filter	Laplace-Filter
			

8.1.7.3 Morphologische Operatoren

In diesem Beispiel wenden Sie morphologische Operatoren auf ein Bild an. Dazu nutzen Sie:

- F_VN_CreateStructuringElement [▶ 1314]
- F_VN_MorphologicalOperator [▶ 1338]

Erklärung

Mit einem morphologischen Operator lassen sich Objekte auf einem Bild nach bestimmten Regeln verändern. Meistens werden morphologische Operatoren auf binäre Bilder angewandt; grundsätzlich ist jedoch auch die Anwendung auf Grauwertbilder möglich. Durch kombinierte Operatoren wie das `OPENING` und das `CLOSING` können z. B. fehlende Teile in Objekten ergänzt und kleine Ausreißer entfernt werden.

Variablen

```
ipElement   :  ITcVnImage;
eSEShape    :  ETcVnStructuringElementShape := TCVN_SES_RECTANGLE;
aSESize     :  ARRAY [0..1] OF UDINT := [29, 29];
eOperator   :  ETcVnMorphologicalOperator := TCVN_MO_OPENING;
```

Programm

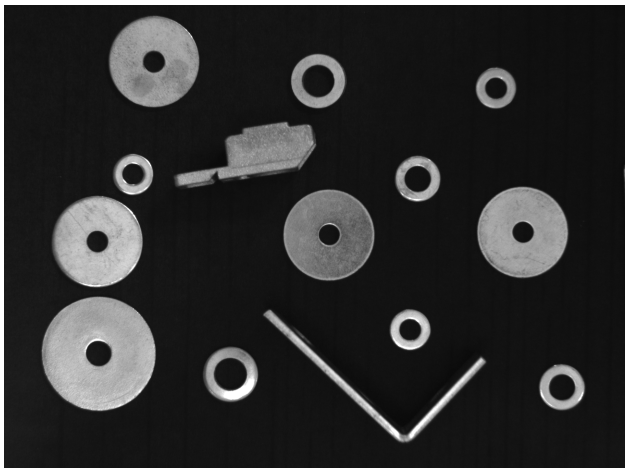
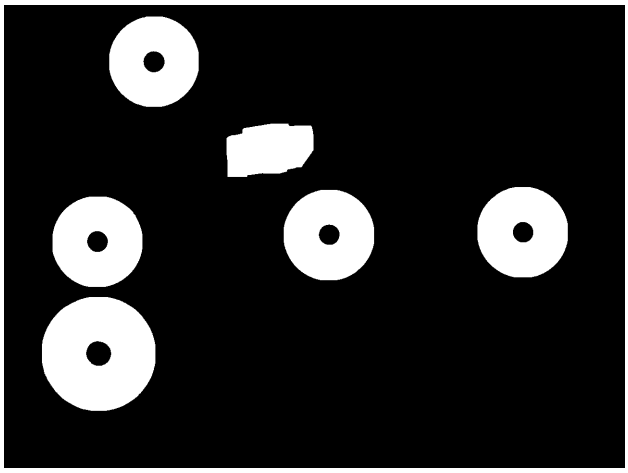
Ein morphologischer Operator benötigt ein Strukturelement, um seinen Einflussbereich für jedes Pixel festzulegen. Sie können dafür aus verschiedenen Formen aus dem Enum ETcVnStructuringElementShape [▶ 229] und in unterschiedlicher Größe auswählen.

```
hr := F_VN_CreateStructuringElement(
    ipStructuringElement    :=  ipElement,
    eShape                  :=  eSEShape,
    nWidth                  :=  aSESize[0],
    nHeight                 :=  aSESize[1],
    hrPrev                  :=  hr);
```

Daraufhin legen Sie eine morphologische Operation mittels des Enums ETcVnMorphologicalOperator [▶ 220] fest und wenden diese mit dem erstellten Strukturelement auf das Eingangsbild an.

```
hr := F_VN_MorphologicalOperator(
    ipSrcImage              :=  ipImageIn,
    ipDestImage             :=  ipImageRes,
    eOperator               :=  eOperator,
    ipStructuringElement    :=  ipElement,
    hrPrev                  :=  hr);
```

Ein morphologisches Öffnen mit einem rechteckigen Strukturelement der Größe [31; 31] und vorherigem Schwellwert erzeugt z. B. folgendes Ergebnis. Es entfernt kleinere bzw. schmalere Objekte aus dem Bild.

Eingangsbild	Ergebnisbild
	

HINWEIS

Hohe Ausführungsdauer

Je nach Größe des Eingangsbilds und des Strukturelements kann die Ausführung des morphologischen Operators sehr lange im Vergleich zu anderen Funktionen dauern. Achten Sie darauf, dass Sie die Zykluszeit entsprechend einstellen und ggfs. einen Watchdog verwenden!

8.1.8 Image Segmentation

- Check Color Range mit RGB-Range [▶ 2897]

8.1.8.1 Check Color Range mit RGB-Range

In diesem Beispiel werden mit Hilfe der Funktion F_VN_CheckColorRange [▶ 1360] rote, grüne und blaue Objekte im Bild gesucht und entsprechend ihrer Farbe beschriftet. Dabei wird die Funktion auf RGB-Bilder angewendet. Es ist auch möglich, die Funktion auf Bilder anderer Farbräume wie zum Beispiel HSV, Lab und BGR anzuwenden, die Parametrierung ist dann entsprechend anzupassen.

Variablen

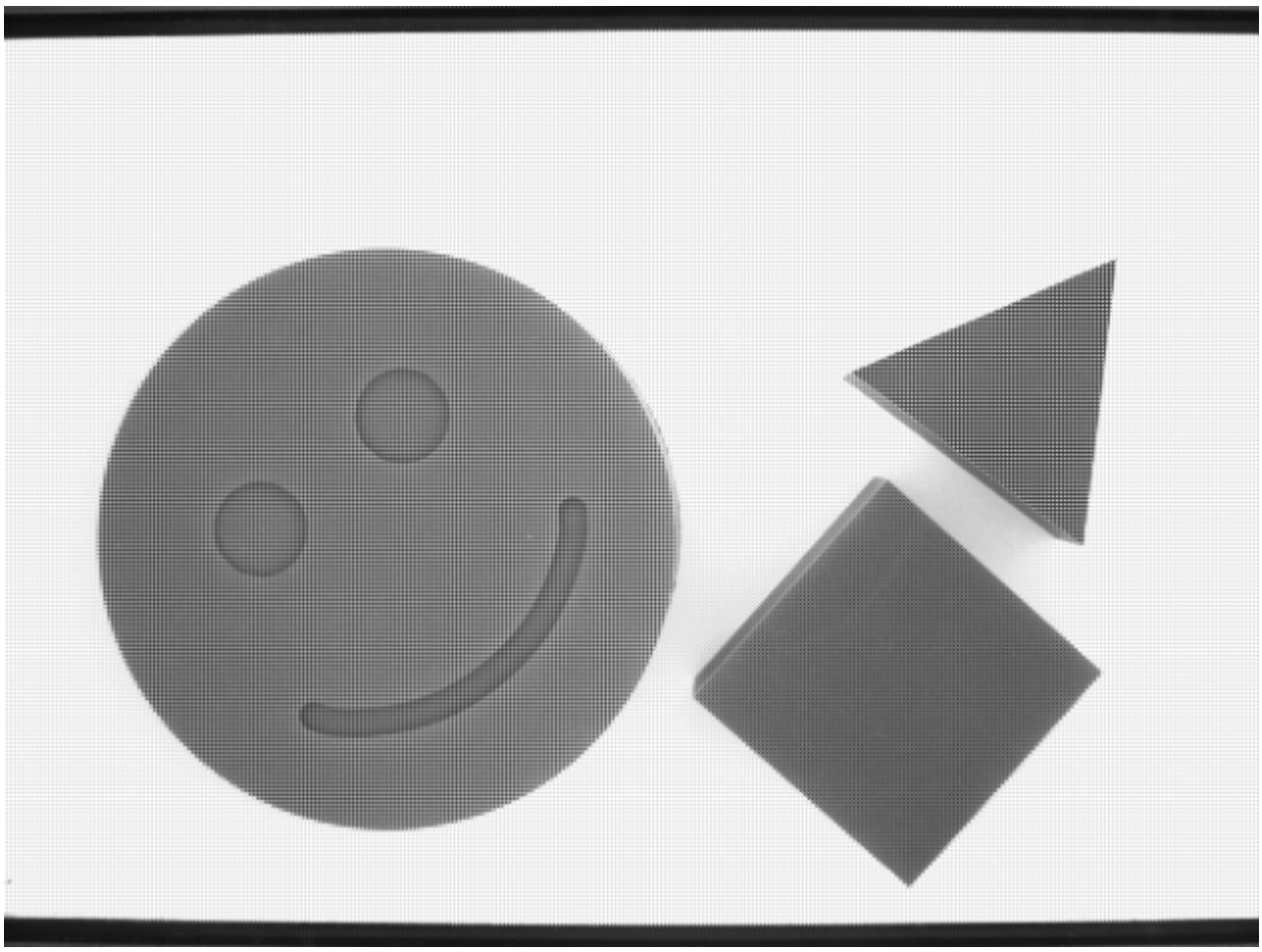
```
hr                  :   HRESULT;

// Images
ipImageIn           :   ITcVnImage;
ipImageInDisp       :   ITcVnDisplayableImage;

ipImageWorkCol      :   ITcVnImage;
ipImageWorkColDisp  :   ARRAY [0..2] OF ITcVnDisplayableImage;

ipImageRes          :   ITcVnImage;
ipImageResDisp      :   ITcVnDisplayableImage;

// Colors
iColor              :   INT;
aColorTxt           :   ARRAY[0..2] OF STRING := [ 'RED', 'GREEN', 'BLUE' ];
aColor              :   ARRAY[0..2] OF TcVnVector4_LREAL := [ [150, 0, 0], [0, 255, 0], [0, 0,
255] ];
aColorRefLow        :   ARRAY[0..2] OF TcVnVector4_LREAL := [ [150, 50, 20], [35, 90, 60], [20, 40,
130] ];
aColorRefUp         :   ARRAY[0..2] OF TcVnVector4_LREAL := [ [255, 120, 100], [100, 200, 140], [60,
160, 255] ];

// Contours
ipContourList       :   ITcVnContainer;
ipIterator          :   ITcVnForwardIterator;
ipContour           :   ITcVnContainer;
fArea               :   LREAL;
aCenter             :   TcVnPoint2_LREAL;
```

Code

```
// Attention: With other images another color space transformation could be necessary
hr := F_VN_ConvertColorSpace(ipImageIn, ipImageRes, TCVN_CST_Bayer_RG_TO_RGB, hr);

FOR iColor := 0 TO 2 DO

   // Apply a "Color-Threshold" on the image
   hr := F_VN_CheckColorRange( ipSrcImage   := ipImageRes,
                               ipDestImage  := ipImageWorkCol,
                               aLowerBounds := aColorRefLow[iColor],
                               aUpperBounds := aColorRefUp[iColor],
                               hrPrev       := hr );

   // Find all objects / contours in the black and white image
   hr := F_VN_FindContours(ipImageWorkCol, ipContourList, hr);
   hr := F_VN_GetForwardIterator(ipContourList, ipIterator, hr);

   // Filter the objects by size and draw the contours
   WHILE SUCCEEDED(hr) AND_THEN ipIterator.CheckIfEnd() <> S_OK DO
      hr := F_VN_GetContainer(ipIterator, ipContour, hr);
      hr := F_VN_IncrementIterator(ipIterator, hr);

      // Filter contours by size
      hr := F_VN_ContourArea(ipContour, fArea, hr);

      IF fArea > 5000 THEN
         // Draw Results into an Image
         hr := F_VN_DrawContours(ipContour, -1, ipImageRes, aColor[iColor], 3, hr);
         hr := F_VN_ContourCenterOfMass(ipContour, aCenter, hr);
         hr := F_VN_PutText(aColorTxt[iColor], ipImageRes, LREAL_TO_UDINT(aCenter[0])-30,
LREAL_TO_UDINT(aCenter[1])+10, TCVN_FT_HERSHEY_PLAIN, 2, aColor[iColor], hr);
      END_IF

   END_WHILE

END_FOR
```

Ergebnis

Das Eingangsbild im Bayer RG Format.



Das Eingangsbild wird mit der Funktion F_VN_ConvertColorSpace [▶ 1270] in ein RGB Bild umgewandelt.

```
hr := F_VN_ConvertColorSpace(ipImageIn, ipImageRes, TCVN_CST_Bayer_RG_TO_RGB, hr);
```

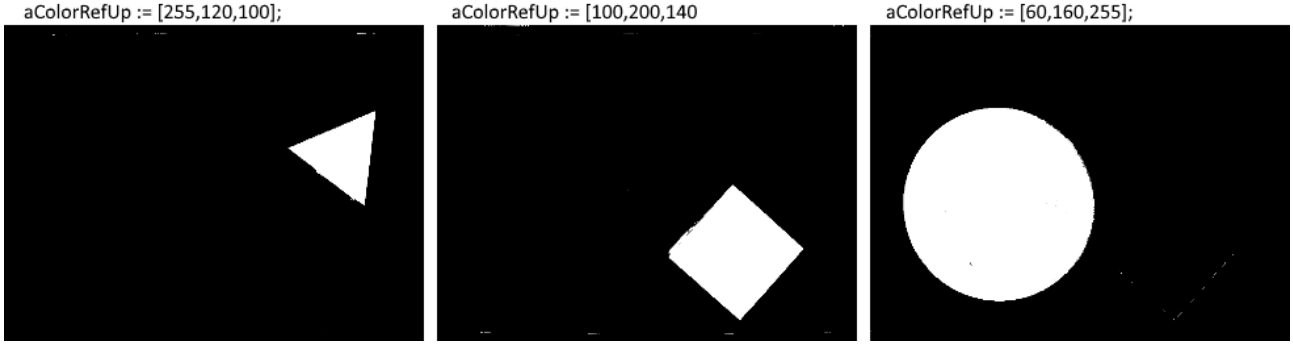
Anschließend kann die Funktion F_VN_CheckColorRange [▶ 1360] mit den jeweiligen oberen und unteren Schwellwerten angewendet werden.

```
hr := F_VN_CheckColorRange(ipImageRes, ipImageWorkCol, aColorRefLow[iColor], aColorRefUp[iColor],
hr);
```

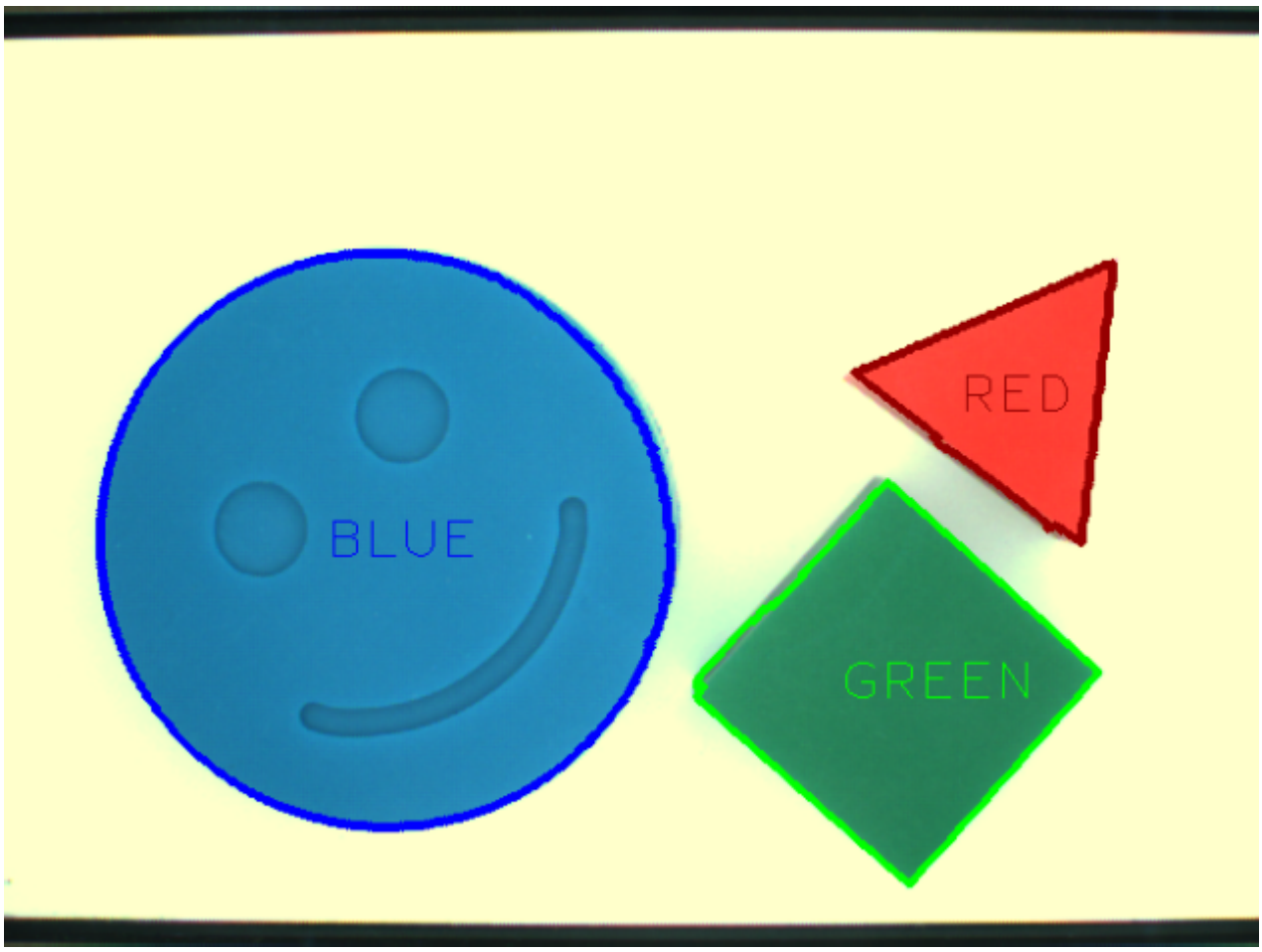
Red
aColorRefLow := [150,50,20];
aColorRefUp := [255,120,100];

Green
aColorRefLow := [35,90,60];
aColorRefUp := [100,200,140

Blue
aColorRefLow := [20,40,130];
aColorRefUp := [60,160,255];



Das Ergebnisbild, nach anschließender Detektion



Ähnliche Beispiele

- Color Similarity mit RGB-Referenz-Farbe [▶ 2887]

8.1.9 Measurement

8.1.9.1 Locate Circular Arc

In diesem Beispiel werden mit Hilfe der Funktion F_VN_LocateCircularArc [▶ 1520] Kreissegmente lokalisiert und vermessen. Als Ergebnis liefert diese Funktion eine TcVnCircularArc-Struktur zurück, die den Mittelpunkt, Radius und die begrenzenden Winkel enthält. Falls vollständige Kreise vermessen werden sollen, verwenden Sie stattdessen die Funktion F_VN_LocateEllipse [▶ 1538].

Die Startposition der Suche sollte in etwa dem erwarteten Kreismittelpunkt des Kreissegments entsprechen, der in den zugehörigen Beispielbildern bei 400, 300 liegt.

Um zu zeigen, dass hier eine ungefähre Position ausreicht, wurde der entsprechende Parameter aCenter im Beispiel auf 420, 310 gesetzt. Es ist also möglich, kleine Positionsänderungen der zu untersuchenden Objekte im Bild zu kompensieren.

Da der tatsächliche Radius in den Beispielbildern bei etwa 200 Pixeln liegt und der Startpunkt absichtlich ungenau gewählt ist, wurde der Suchradius fRadius auf 270 festgelegt. Im konkreten Beispiel hätten auch etwa 230 Pixel ausgereicht, jedoch ist es in der Praxis immer gut, noch einen zusätzlichen Sicherheitspuffer zu haben, z. B. falls die Position des Objekts doch mal stärker abweicht als erwartet.

Für die Beispielbilder muss die Suchrichtung so gewählt werden, dass die Objekte in allen Bildern korrekt gefunden werden. Eine gute Richtung wäre unten links, also etwa 2.356 rad (135°). Um zu zeigen, dass diese Richtung wiederum nicht genau stimmen muss, wurde fDirection im Beispiel auf 2.1 rad (120.3°) gesetzt.

Des Weiteren muss noch definiert werden, dass ausgehend vom Startpunkt ein Übergang von hell nach dunkel gesucht wird. Zudem muss der minimale Kontrast spezifiziert werden, ab welchem die Funktion einen Intensitätsunterschied zwischen benachbarten Pixeln als Kante erkennt.

Für den Fall, dass die erläuterten Parameter nicht ausreichen um gute Ergebnisse zu erzielen, bietet die Experten-Funktion F_VN_LocateCircularArcExp [▶ 1523] eine Reihe weiterer Parameter, die in diesem Beispiel jedoch nicht betrachtet werden.

Variablen

```
hr               : HRESULT;

ipImageIn        : ITcVnImage;
ipImageInDisp    : ITcVnDisplayableImage;
ipImageRes       : ITcVnImage;
ipImageResDisp   : ITcVnDisplayableImage;

// result
stArc            : TcVnCircularArc;

// input parameters (to specify where to start searching for the circular arc)
aCenter          : TcVnPoint2_REAL := [420, 310];
fRadius          : REAL := 270;
fDirection       : LREAL := 2.1;

// drawing
aColor           : TcVnVector4_LREAL := [0, 175, 0];
sText            : STRING(255);
```

Code

```
hr := F_VN_LocateCircularArc(
                    ipSrcImage          :=   ipImageIn,
                    stCircularArc       :=   stArc,
                    aCenterPoint        :=   aCenter,
                    fSearchRadius       :=   fRadius,
                    fArcDirectionRad    :=   fDirection,
                    eEdgeDirection      :=   TCVN_ED_LIGHT_TO_DARK,
                    fMinStrength        :=   100,
                    hrPrev              :=   hr);

// Draw result for visualization
hr := F_VN_ConvertColorSpace(ipImageIn, ipImageRes, TCVN_CST_GRAY_TO_RGB, hr);
hr := F_VN_DrawCircularArc(stArc, ipImageRes, aColor, 2, hr);
sText := CONCAT(CONCAT(CONCAT('Center ', REAL_TO_STRING(stArc.aCenter[0])), ', '),
REAL_TO_STRING(stArc.aCenter[1]));
hr := F_VN_PutText(sText, ipImageRes, 420, 25, TCVN_FT_HERSHEY_SIMPLEX, 0.7, aColor, hr);
sText := CONCAT('Radius ', REAL_TO_STRING(stArc.fRadius));
hr := F_VN_PutText(sText, ipImageRes, 420, 50, TCVN_FT_HERSHEY_SIMPLEX, 0.7, aColor, hr);
sText := CONCAT('Angle start ', REAL_TO_STRING(stArc.fStartAngle * 180 / LREAL_TO_REAL(PI)));
hr := F_VN_PutText(sText, ipImageRes, 420, 75, TCVN_FT_HERSHEY_SIMPLEX, 0.7, aColor, hr);
sText := CONCAT('Angle end ', REAL_TO_STRING(stArc.fEndAngle * 180 / LREAL_TO_REAL(PI)));
hr := F_VN_PutText(sText, ipImageRes, 420, 100, TCVN_FT_HERSHEY_SIMPLEX, 0.7, aColor, hr);

// Display source and result image
hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, S_OK);
hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, S_OK);
```

Ergebnisse

Zur Visualisierung wird das gefundene Kreissegment mit Hilfe der Funktion F_VN_DrawCircularArc [▶ 1035] in das Ergebnisbild eingezeichnet und zusätzlich noch die zugrundeliegenden Zahlenwerte als Text dargestellt.

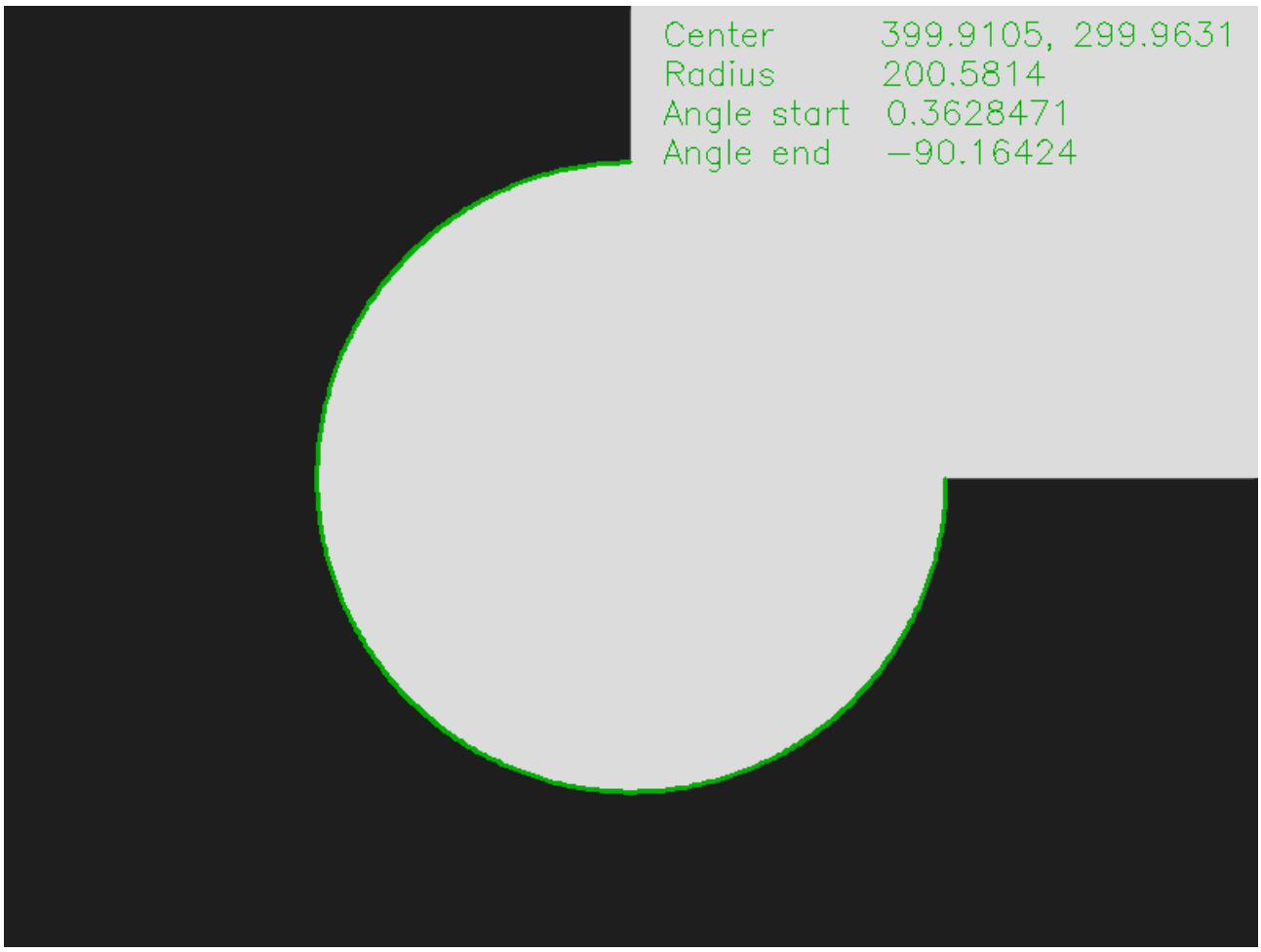


Abb. 28: LocateCircularArc_3QuarterCircle_Result

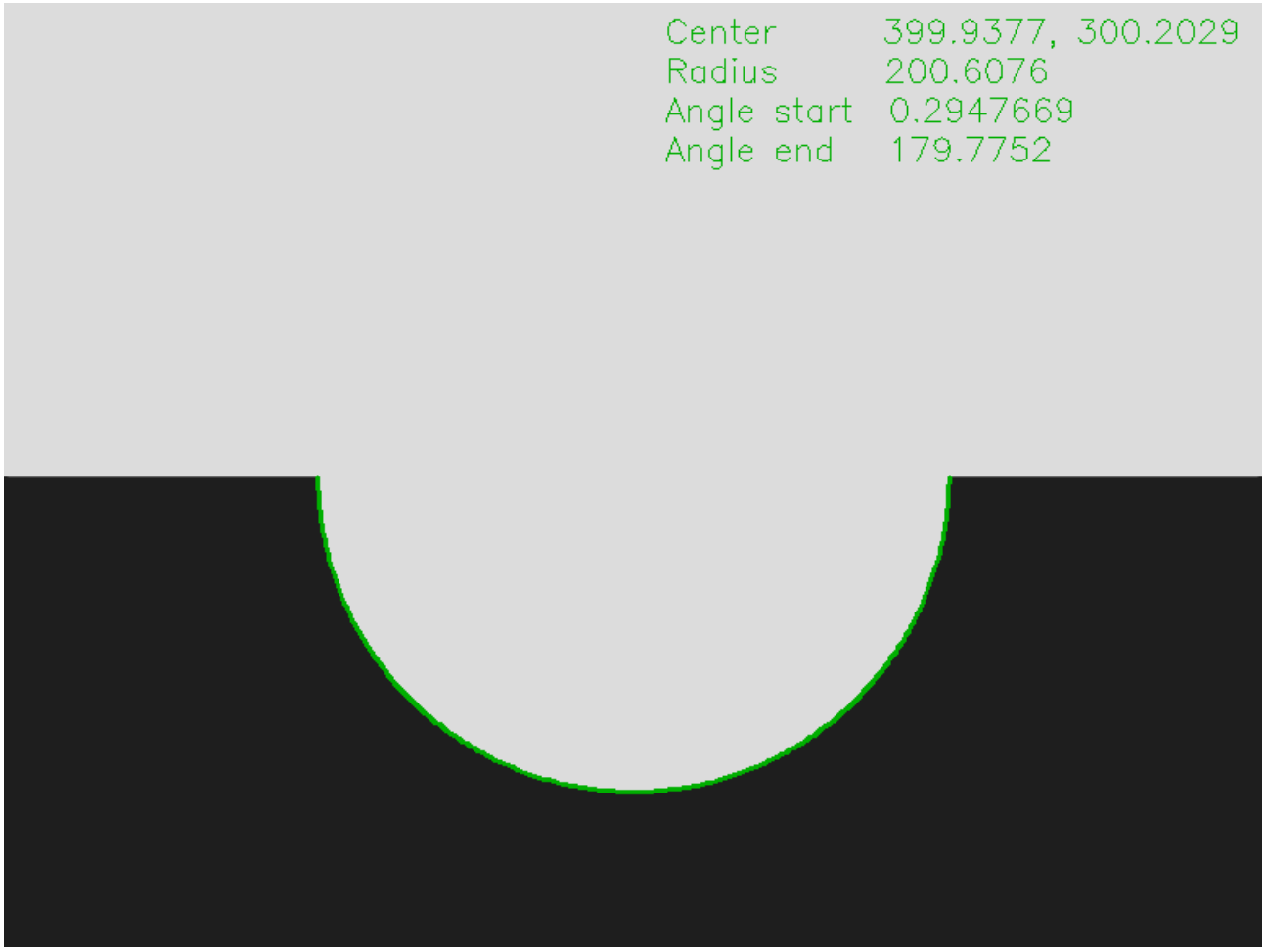


Abb. 29: LocateCircularArc_HalfCircle_Result

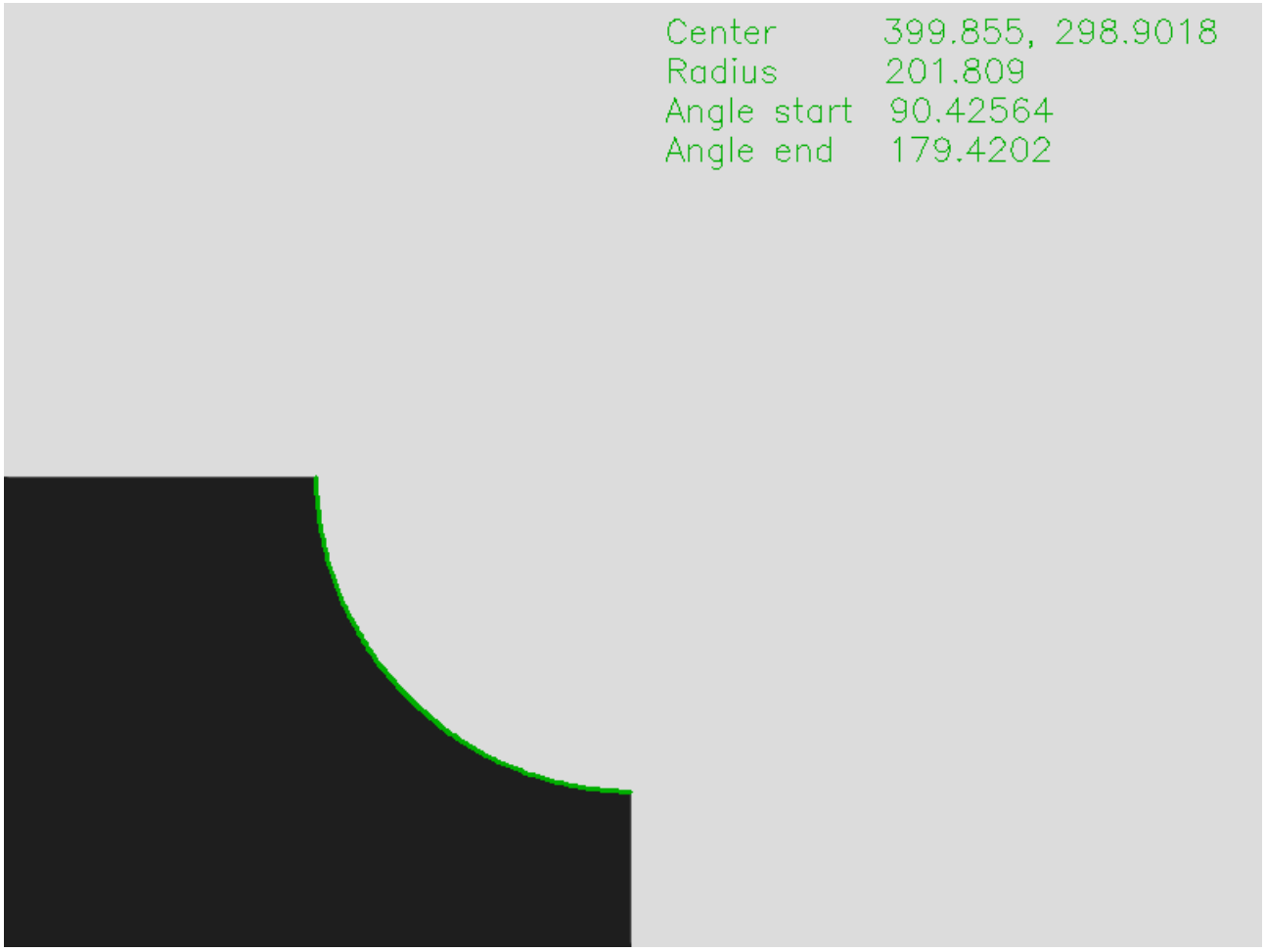


Abb. 30: LocateCircularArc_QuarterCircle_Result

Ähnliche Beispiele

- Locate Edge [▶ 2904]
- Locate Ellipse [▶ 2909]
- Measure Angle Between Edges [▶ 2914]

8.1.9.2 Locate Edge

In diesem Beispiel wird

- mit Hilfe der Funktion F_VN_LocateEdgeExp [▶ 1529] eine Kante in einem definierten Suchfenster lokalisiert
- und die Ausführungszeit mit einem Watchdog überwacht und bei Bedarf begrenzt.

Erläuterung

Innerhalb eines Suchfensters können Kanten mit der Funktion F_VN_LocateEdge [▶ 1526] lokalisiert werden. Zur einfacheren Anwendung sind einige Parameter dabei festgesetzt. Alternativ steht mit der hier verwendeten Funktion F_VN_LocateEdgeExp [▶ 1529] ein voller Zugriff auf alle Parameter zur Verfügung. Dieses Beispiel soll dabei helfen die einzelnen Parameter besser zu verstehen. Probieren Sie daher neben der Standardkonfiguration auch Konfigurationsänderungen aus und betrachten Sie die Auswirkungen auf das Kantenergebnis und auf die Verarbeitungszeit. Eine Auswahl von Konfigurationsänderungen inkl. Beschreibung finden Sie unter Ergebnisse.

Variablen

```
hr              : HRESULT;
hrFunc          : HRESULT;

ipImageIn       : ITcVnImage;
ipImageInDisp   : ITcVnDisplayableImage;
ipImageRes      : ITcVnImage;
ipImageResDisp  : ITcVnDisplayableImage;

// result
ipEdgePoints    : ITcVnContainer;

// parameters
aStartPoint     : TcVnPoint2_REAL := [850, 400];
aEndPoint       : TcVnPoint2_REAL := [550, 400];
eDirection      : ETcVnEdgeDirection := TCVN_ED_DARK_TO_LIGHT;
fMinStrength    : REAL := 50;
nSearchLines    : UDINT := 31;
fSearchLineDist : REAL := 1;
nMaxThickness   : UDINT := 7;
nSubpixIter     : UDINT := 10;
eAlgorithm      : ETcVnEdgeDetectionAlgorithm := TCVN_EDA_INTERPOLATION;
fAvgStrength    : REAL;

// Watchdog
hrWD            : HRESULT;
tStop           : DINT := 15000;
tRest           : DINT;
nFraction       : UDINT;

// drawing
aLine           : TcVnVector4_LREAL;
aColorGreen     : TcVnVector4_LREAL := [0, 175, 0];
aColorBlue      : TcVnVector4_LREAL := [0, 0, 255];
aColorRed       : TcVnVector4_LREAL := [255, 0, 0];
sText           : STRING(255);
```

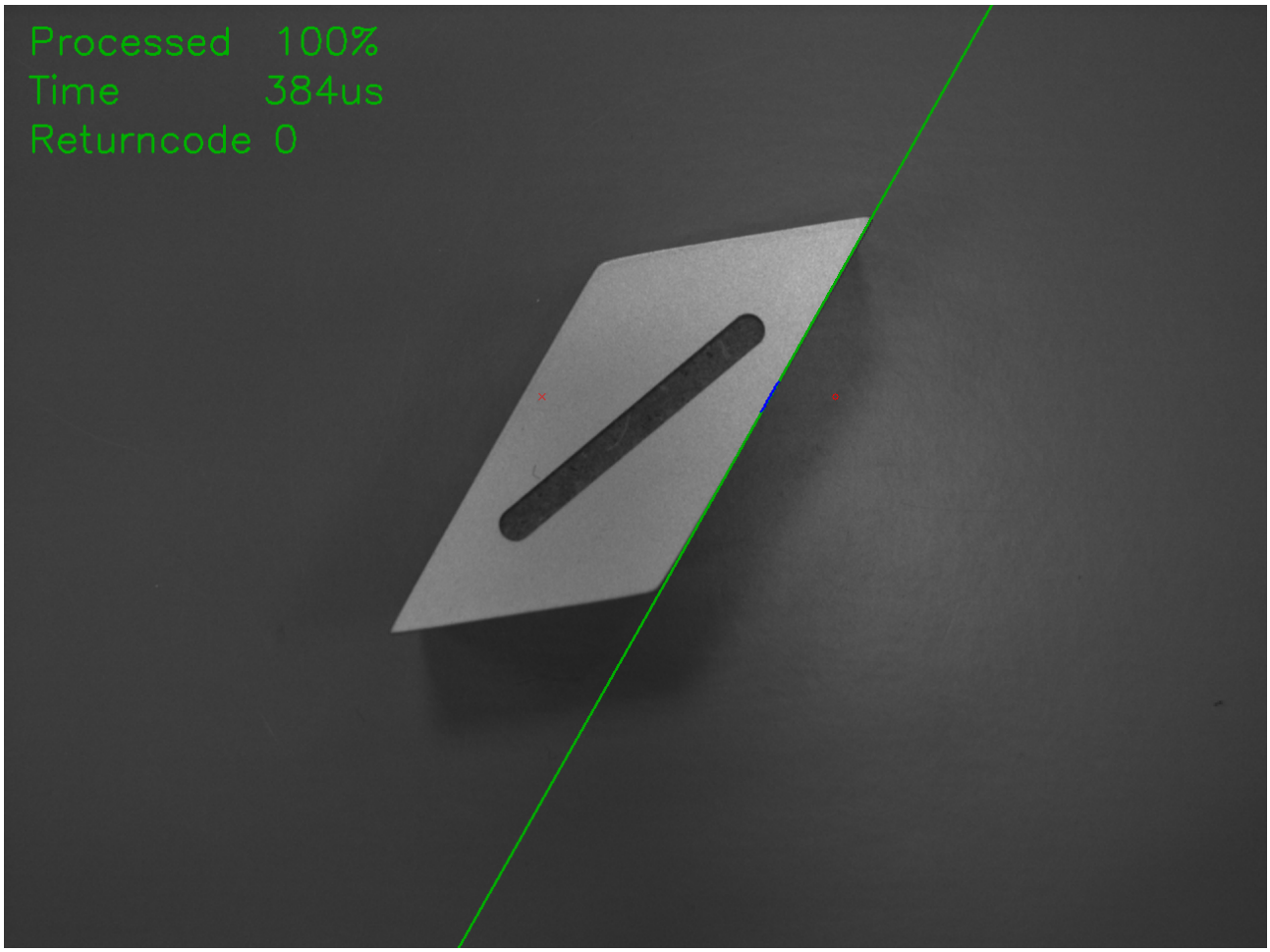
Code

```
hrWD := F_VN_StartRelWatchdog(tStop, hr);
   hrFunc := F_VN_LocateEdgeExp(
                  ipSrcImage             :=   ipImageIn,
                  ipEdgePoints           :=   ipEdgePoints,
                  aStartPoint            :=   aStartPoint,
                  aEndPoint              :=   aEndPoint,
                  eEdgeDirection         :=   eDirection,
                  fMinStrength           :=   fMinStrength,
                  nSearchLines           :=   nSearchLines,
                  fSearchLineDist        :=   fSearchLineDist,
                  nMaxThickness          :=   nMaxThickness,
                  nSubpixelsIterations   :=   nSubpixIter,
                  fApproxPrecision       :=   0.0001,
                  eAlgorithm             :=   eAlgorithm,
                  hrPrev                 :=   hr,
                  fAvgStrength           =>   fAvgStrength);
hrWD := F_VN_StopWatchdog(hrWD, nFractionProcessed=>nFraction, tRest=>tRest);

// Draw result for visualization
hr := F_VN_ConvertColorSpace(ipImageIn, ipImageRes, TCVN_CST_GRAY_TO_RGB, hr);
sText := CONCAT(CONCAT('Processed ', UDINT_TO_STRING(nFraction)), '%');
hr := F_VN_PutTextExp(sText, ipImageRes, 25, 50, TCVN_FT_HERSHEY_SIMPLEX, 1.3, aColorGreen, 2,
TCVN_LT_8_CONNECTED, FALSE, hr);
sText := CONCAT(CONCAT('Time ', DINT_TO_STRING(tStop - tRest)), 'us');
hr := F_VN_PutTextExp(sText, ipImageRes, 25, 100, TCVN_FT_HERSHEY_SIMPLEX, 1.3, aColorGreen, 2,
TCVN_LT_8_CONNECTED, FALSE,hr);
sText := CONCAT('Returncode ', DINT_TO_STRING(hrFunc));
hr := F_VN_PutTextExp(sText, ipImageRes, 25, 150, TCVN_FT_HERSHEY_SIMPLEX, 1.3, aColorGreen, 2,
TCVN_LT_8_CONNECTED, FALSE,hr);
hr := F_VN_DrawPoint(REAL_TO_UDINT(aStartPoint[0]), REAL_TO_UDINT(aStartPoint[1]), ipImageRes,
TCVN_DS_CIRCLE, aColorRed, hr);
hr := F_VN_DrawPoint(REAL_TO_UDINT(aEndPoint[0]), REAL_TO_UDINT(aEndPoint[1]), ipImageRes,
TCVN_DS_X, aColorRed, hr);
hr := F_VN_FitLine(ipEdgePoints, aLine, hr);
hr := F_VN_DrawLine_TcVnVector4_LREAL(aLine, ipImageRes, aColorGreen, 2, hr);
hr := F_VN_DrawPointsExp(ipEdgePoints, ipImageRes, TCVN_DS_PLUS, aColorBlue, 1, 1,
TCVN_LT_8_CONNECTED, hr);
```

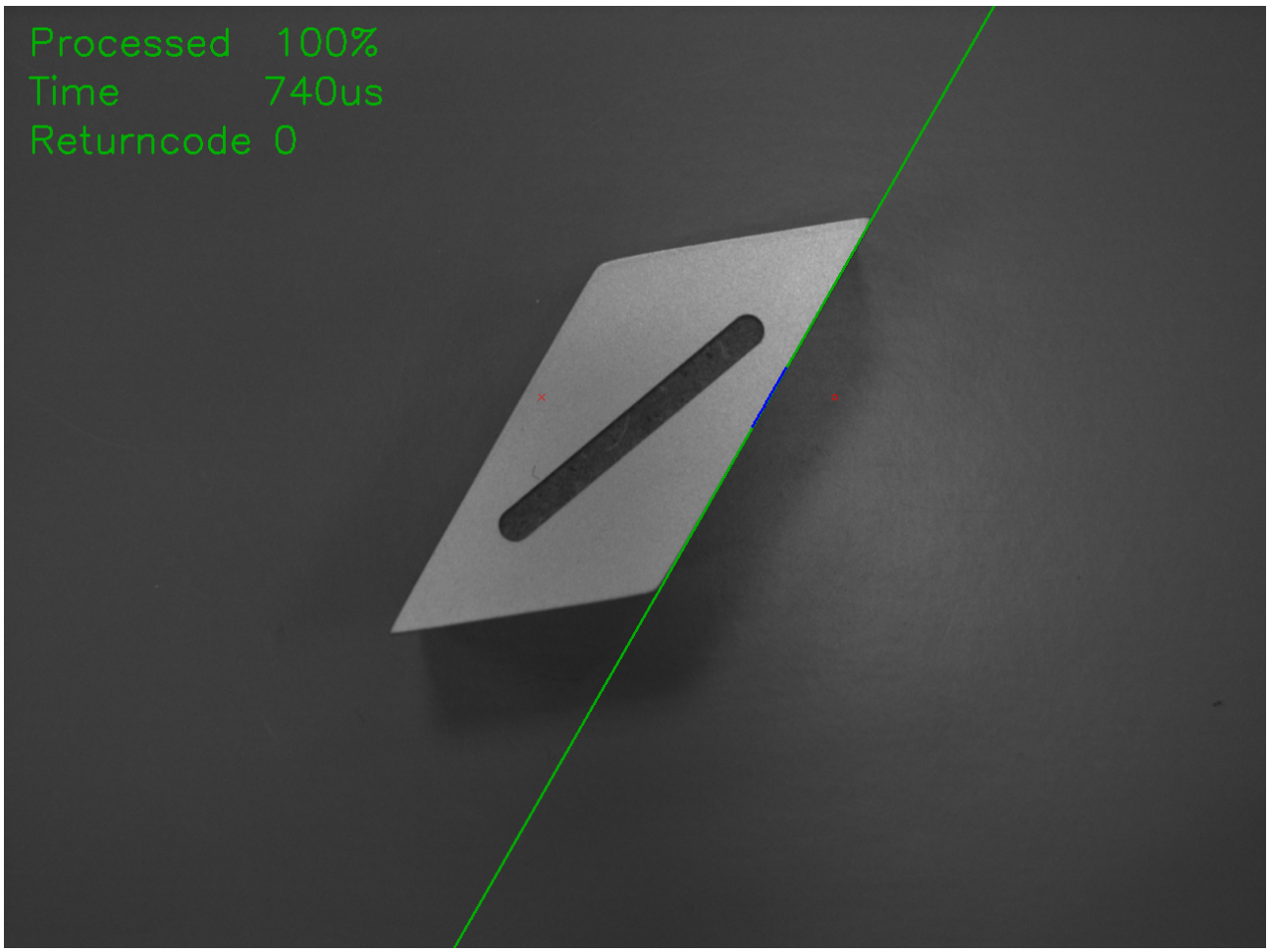
```
// Display source and result image
hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, S_OK);
hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, S_OK);
```

Ergebnisse

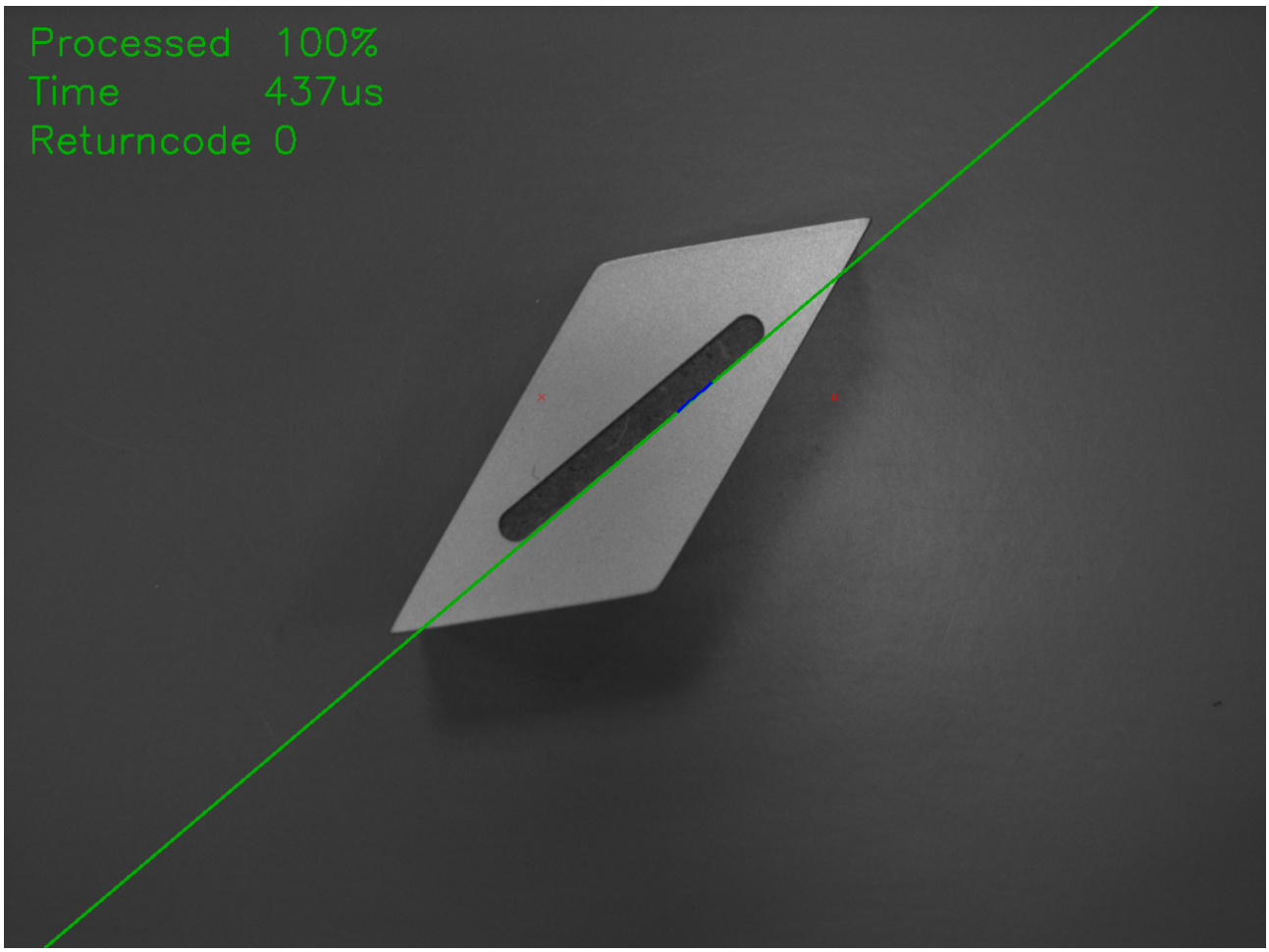
Zur Visualisierung werden die gefundenen Kantenpunkte in blau und eine Ausgleichsgerade in grün in das Ausgabebild gezeichnet. Zudem werden der Startpunkt (roter Kreis) und der Endpunkt (rotes X) eingezeichnet. Oben links werden der Bearbeitungsanteil, die benötigte Ausführungszeit und der Rückgabewert der Funktion angezeigt.
Für die vorgegebenen Parameter sieht das Ergebnis folgendermaßen aus:



Wird nun die Anzahl der Suchlinien mit dem Parameter `nSearchLines` von 31 auf 61 erhöht, verdoppelt sich auch die benötigte Rechenzeit. Die Ausgleichsgerade wird im Gegenzug aber auch genauer:



Soll anstelle der Außenkante des Bauteils die Kante der Fräsung auf dem Bauteil gefunden werden, muss der Parameter `eDirection` auf `TCVN_ED_LIGHT_TO_DARK` geändert werden. Dadurch wird der äußere Kantenübergang von dunkel nach hell ignoriert und stattdessen die nachfolgende Kante mit dem Übergang hell nach dunkel gefunden:



Ausführungszeit reduzieren

Hierbei fällt auf, dass die Ausführungszeit trotz ansonsten identischer Parameter höher ist als beim Lokalisieren der äußeren Kante. Das liegt daran, dass die gefundene Kante weiter vom Startpunkt entfernt liegt.

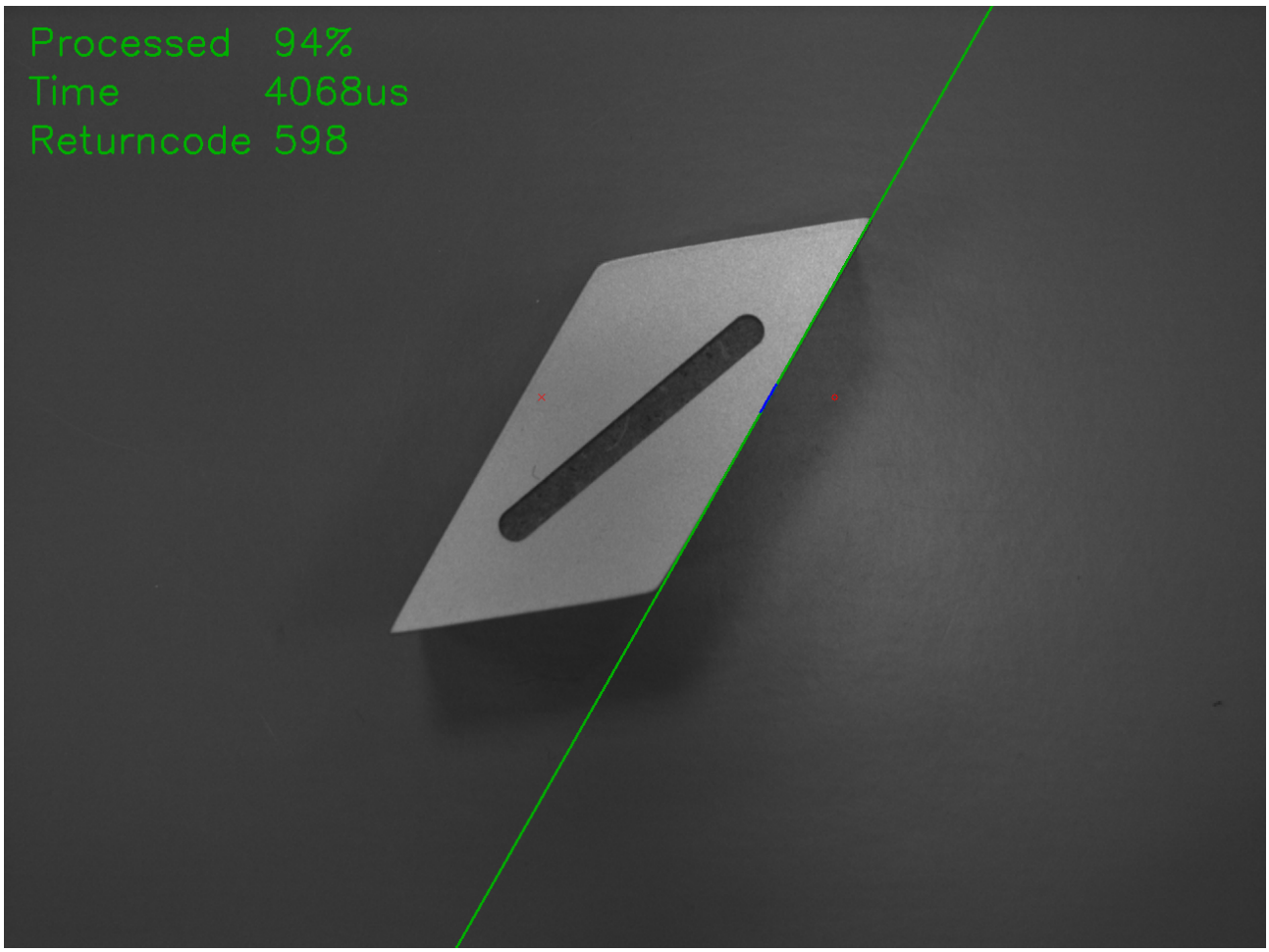
Im Umkehrschluss heißt das auch, dass die Ausführungszeit bei einem festen Suchfenster je nach Position des Bauteils im Bild etwas schwankt, was sich auch gut bei den drei Beispielbildern beobachten lässt.

Ein noch größeres Schwanken kann bei den beiden Approximierungsalgorithmen auftreten, da diese unterschiedlich viele Iterationen benötigen, um die Modellparameter mittels der umliegenden Pixelintensitäten zu approximieren.

Generell werden weniger Iterationen benötigt, wenn die Suchlinien des Suchfensters möglichst orthogonal auf die Kante treffen. Falls die Position und Orientierung der Objekte im Bild keinen oder nur geringen Schwankungen unterliegen, richten Sie das Suchfenster entsprechend aus, um kürzere Ausführungszeiten zu erreichen. Zudem lässt sich die maximal benötigte Zeit reduzieren, indem die maximale Anzahl an Iterationen verringert wird, was aber ggf. auch ungenauere Ergebnisse liefert.

Watchdog nutzen

Zudem kann die Zeit auch mit einem Watchdog von außen begrenzt werden, der die Ausführung der Funktion bei Bedarf abbricht und die vorhandenen Teilergebnisse zurückliefert. Wird zum Beispiel der Algorithmus auf `TCVN_EDA_APPROX_ERF` geändert und die maximale Anzahl Iterationen auf 60 festgelegt, benötigt die Ausführung für das Bild LocateEdge2.bmp (Außenkante orthogonal zu den Suchlinien) ziemlich genau 4ms (abhängig von der verbauten CPU), wohingegen die anderen beiden Bilder 4.3 bzw. 4.4 ms benötigen. Setzt man nun den Parameter `tStop` auf 4000, wird die Ausführung der Funktion nach 4ms zum nächst möglichen Zeitpunkt gestoppt und die bis dahin vorhandenen Teilergebnisse zurückgeliefert. Daher wird die Funktion auf dem Bild LocateEdge2.bmp weiterhin zu 100% bearbeitet, wohingegen bei den anderen beiden Bildern (hier LocateEdge3.bmp) vorzeitig abgebrochen wird:



Da nur an bestimmten Punkten im Algorithmus abgebrochen werden kann und die Teilergebnisse (bisher gefundene Kantenpunkte auf den Suchlinien) zurückgeliefert werden, wird die Funktion nicht nach exakt 4000µs beendet, sondern benötigt etwas länger, was es bei der Wahl der Abbruchzeit zu berücksichtigen gilt. Die maximal zusätzlich benötigte Zeit ist generell abhängig von dem Algorithmus und der Parametrierung.

Trotz des Abbruchs reichen die Teilergebnisse aus, um das gewünschte Ergebnis zu erzielen, da lediglich die Resultate von ein oder zwei Suchlinien fehlen.

Ähnliche Beispiele

- Locate Circular Arc [▶ 2900]
- Locate Ellipse [▶ 2909]
- Measure Angle Between Edges [▶ 2914]

8.1.9.3 Locate Ellipse

In diesem Beispiel wird

- mit Hilfe der Funktion F_VN_LocateEllipseExp [▶ 1541] ein kreisförmiges Objekt in einem definierten Suchbereich lokalisiert
- und die Ausführungszeit mit einem Watchdog überwacht und bei Bedarf begrenzt.

Erläuterung

Für ein Grundverständnis der Vermessungsfunktionen und deren Parameter sehen Sie sich zuerst das Locate Edge Beispiel [▶ 2904] an. Die Funktion F_VN_LocateEllipseExp [▶ 1541] unterscheidet sich von der Funktion F_VN_LocateEdgeExp [▶ 1526] dadurch, dass sie ein kreisförmiges statt rechteckiges Suchfenster

verwendet. In diesem Beispiel ist ein großes, ungenaues Suchfenster parametriert, da sich in diesem Beispiel die Position der kreisförmigen Objekte etwas ändert. So können mit dem gleichen Suchfenster in allen Bildern die Kreise bestimmt werden.

Mit dem Parameter `bInvertSearchDirection` - der in diesem Beispiel auf `TRUE` gesetzt wurde - ist definiert worden, dass die Suchrichtung umgekehrt wird, d. h. von außen in Richtung `aCenter`. Das ist notwendig, da sich in den Beispielbildern innerhalb der gesuchten Ellipse ebenfalls Kanten befinden.

Da in diesem Beispiel die Objekte dunkler sind als der Hintergrund, wurde die Suchrichtung auf `TCVN_ED_LIGHT_TO_DARK` festgelegt.

Der Parameter `nMaxThickness`, der definiert, innerhalb von wie vielen Pixeln `fMinStrength` erreicht werden muss, wurde auf den Wert „`7`“ gesetzt.

Variablen

```
hr               : HRESULT;
hrFunc           : HRESULT;

ipImageIn        : ITcVnImage;
ipImageInDisp    : ITcVnDisplayableImage;
ipImageRes       : ITcVnImage;
ipImageResDisp   : ITcVnDisplayableImage;

// result
stEllipse        : TcVnRotatedRectangle;
ipContourPoints  : ITcVnContainer;

// input parameters
aCenter          : TcVnPoint2_REAL := [650, 400];
fRadius          : REAL := 300;
fMinRadius       : REAL := 100;
fMinStrength     : REAL := 30;
nSubPixIter      : UDINT := 10;
nSearchLines     : UDINT := 40;
eAlgorithm       : ETcVnEdgeDetectionAlgorithm := TCVN_EDA_INTERPOLATION;

// Watchdog
hrWD             : HRESULT;
tStop            : DINT := 15000;
tRest            : DINT;
nFraction        : UDINT;

// drawing
aColorRed        : TcVnVector4_LREAL := [200, 0, 0];
aColorGreen      : TcVnVector4_LREAL := [0, 175, 0];
sText            : STRING(255);
```

Code

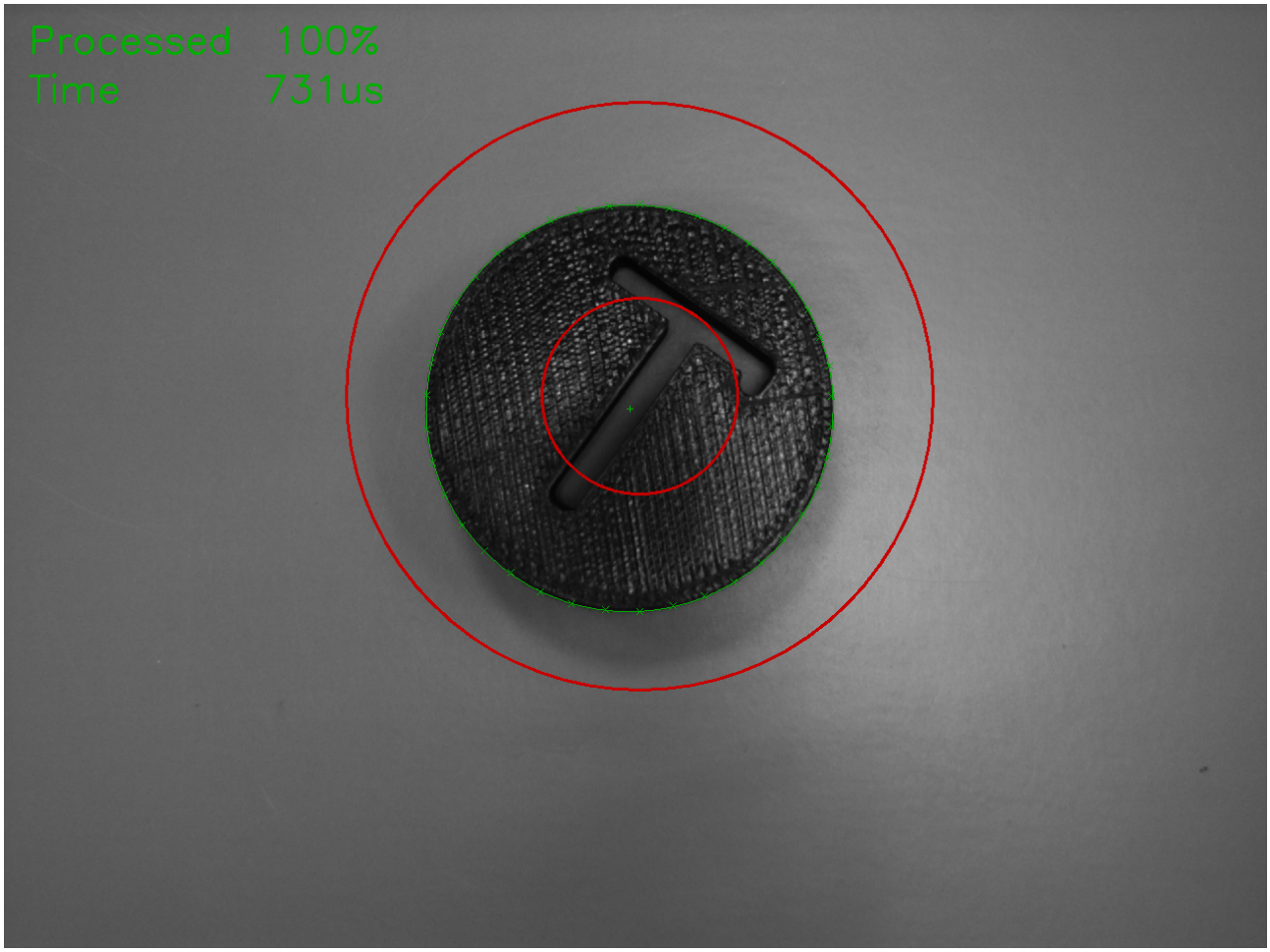
```
hrWD := F_VN_StartRelWatchdog(tStop, hr);
   hrFunc := F_VN_LocateEllipseExp(
             ipSrcImage                :=   ipImageIn,
             stEllipse                 :=   stEllipse,
             aCenter                   :=   aCenter,
             fSearchRadius             :=   fRadius,
             eEdgeDirection            :=   TCVN_ED_LIGHT_TO_DARK,
             fMinStrength              :=   fMinStrength,
             nMaxThickness             :=   7,
             bInvertSearchDirection    :=   TRUE,
             fMinSearchRadius          :=   fMinRadius,
             nSubpixelsIterations      :=   nSubpixIter,
             nSearchLines              :=   nSearchLines,
             fApproxPrecision          :=   0.0001,
             eAlgorithm                :=   eAlgorithm,
             ipContourPoints           :=   ipContourPoints,
             hrPrev                    :=   hr);
hrWD := F_VN_StopWatchdog(hrWD, nFractionProcessed=>nFraction, tRest=>tRest);

// Draw result for visualization
hr := F_VN_ConvertColorSpace(ipImageIn, ipImageRes, TCVN_CST_GRAY_TO_RGB, hr);
sText := CONCAT(CONCAT('Processed ', UDINT_TO_STRING(nFraction)), '%');
hr := F_VN_PutTextExp(sText, ipImageRes, 25, 50, TCVN_FT_HERSHEY_SIMPLEX, 1.3, aColorGreen, 2,
TCVN_LT_8_CONNECTED, FALSE, hr);
sText := CONCAT(CONCAT('Time ', DINT_TO_STRING(tStop - tRest)), 'us');
hr := F_VN_PutTextExp(sText, ipImageRes, 25, 100, TCVN_FT_HERSHEY_SIMPLEX, 1.3, aColorGreen, 2,
```

```
TCVN_LT_8_CONNECTED, FALSE,hr);
hr := F_VN_DrawCircle(REAL_TO_UDINT(aCenter[0]), REAL_TO_UDINT(aCenter[1]),
REAL_TO_UDINT(fMinRadius), ipImageRes, aColorRed, 2, hr);
hr := F_VN_DrawCircle(REAL_TO_UDINT(aCenter[0]), REAL_TO_UDINT(aCenter[1]), REAL_TO_UDINT(fRadius),
ipImageRes, aColorRed, 2, hr);
hr := F_VN_DrawPoints(ipContourPoints, ipImageRes, TCVN_DS_X, aColorGreen, hr);
hr := F_VN_DrawPoint(REAL_TO_UDINT(stEllipse.aCenter[0]), REAL_TO_UDINT(stEllipse.aCenter[1]),
ipImageRes, TCVN_DS_PLUS, aColorGreen, hr);
hr := F_VN_DrawEllipse(stEllipse, ipImageRes, aColorGreen, 1, hr);

// Display source and result image
hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, S_OK);
hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, S_OK);
```

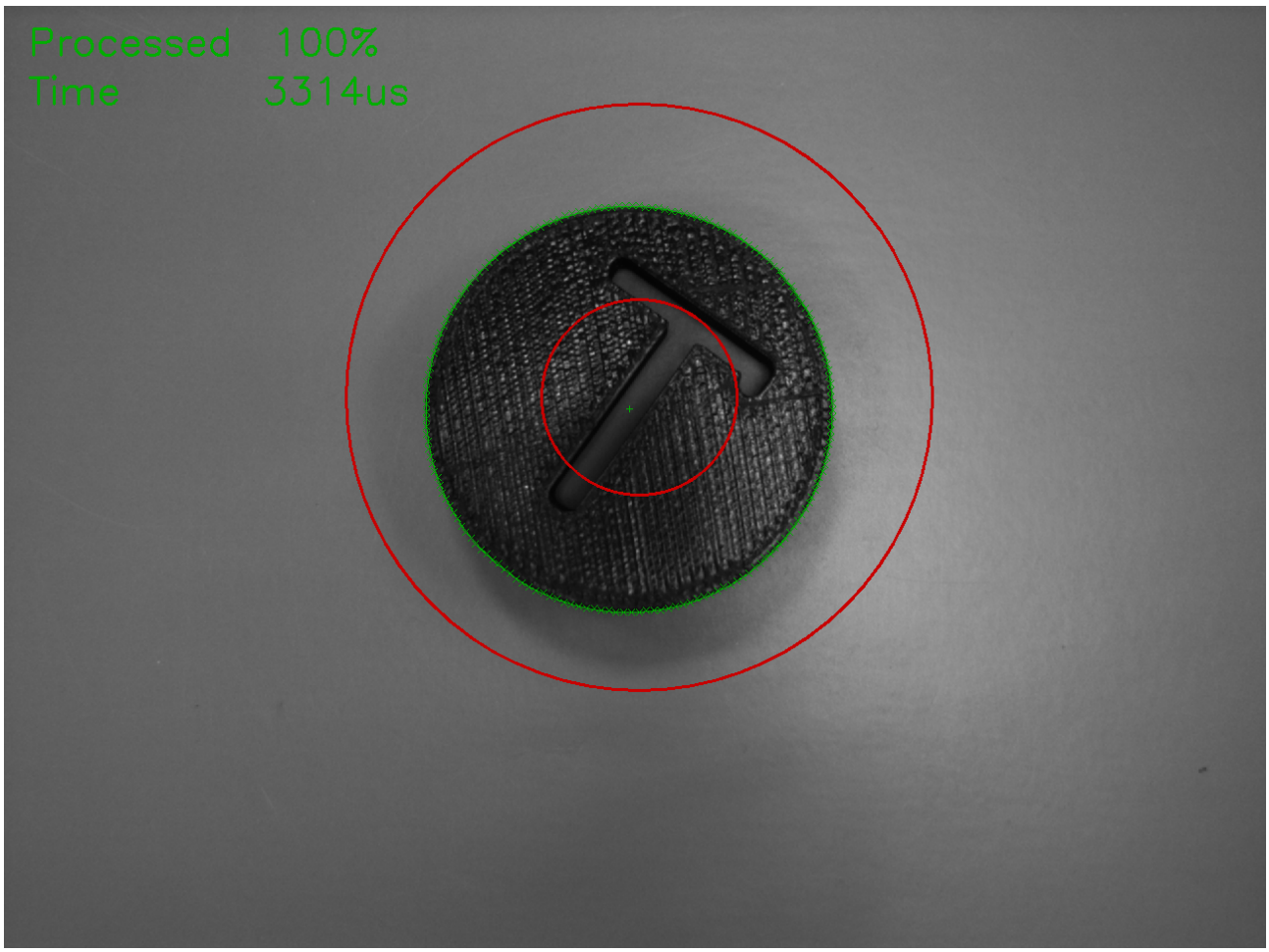
Ergebnisse

Die Funktion liefert die `TcVnRotatedRectangle`-Struktur `stEllipse` zurück, die die gefundene Ellipse anhand ihres Mittelpunkts `aCenter`, ihrer Höhe und Breite `stSize.fHeight` bzw. `stSize.fWidth` und ihres Winkels `fAngle` definiert.
Optional kann zusätzlich ein Kantenpunkt pro Suchlinie `ipContourPoints` zurückgegeben werden, aus denen die Ellipse approximiert wurde.

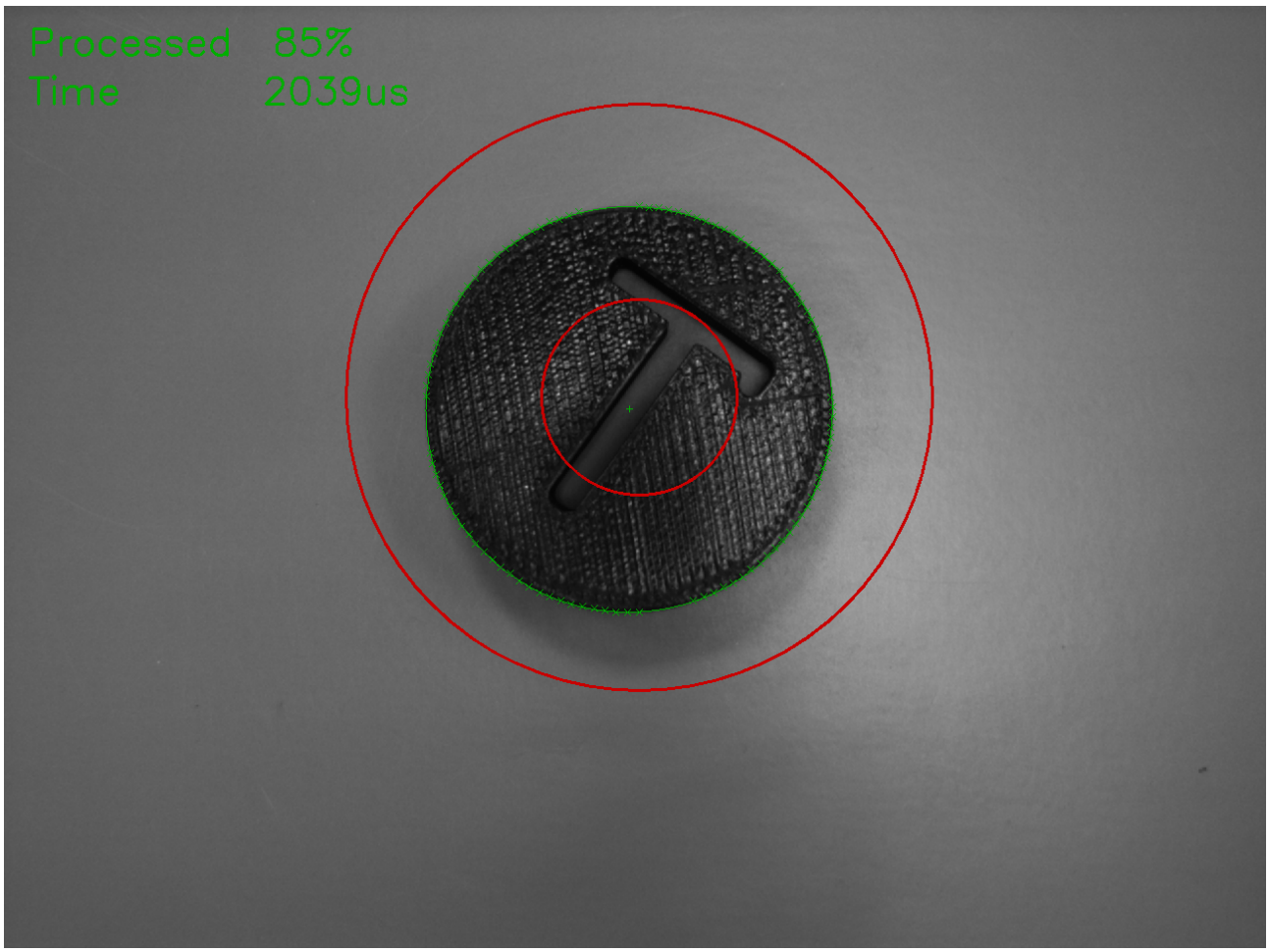
Zur Visualisierung wird zunächst der Suchbereich durch 2 rote Kreise um den vorgegebenen Mittelpunkt ins Bild eingezeichnet. Die gefundene Ellipse wird als grüne Linie eingezeichnet und der zugehörige Mittelpunkt als grünes „+“. Die einzelnen Messpunkte aus `ipContourPoints` werden als grüne x eingezeichnet. Zudem stehen oben links der Bearbeitungsanteil und die benötigte Rechenzeit. Für die vorgegebenen Parameter sieht das Ergebnis für das Eingangsbild *LocateEllipse_T.bmp* folgendermaßen aus:



Erhöht man nun die Anzahl der Suchlinien von 40 auf 180, wird die benötigte Rechenzeit wie erwartet linear dazu erhöht:

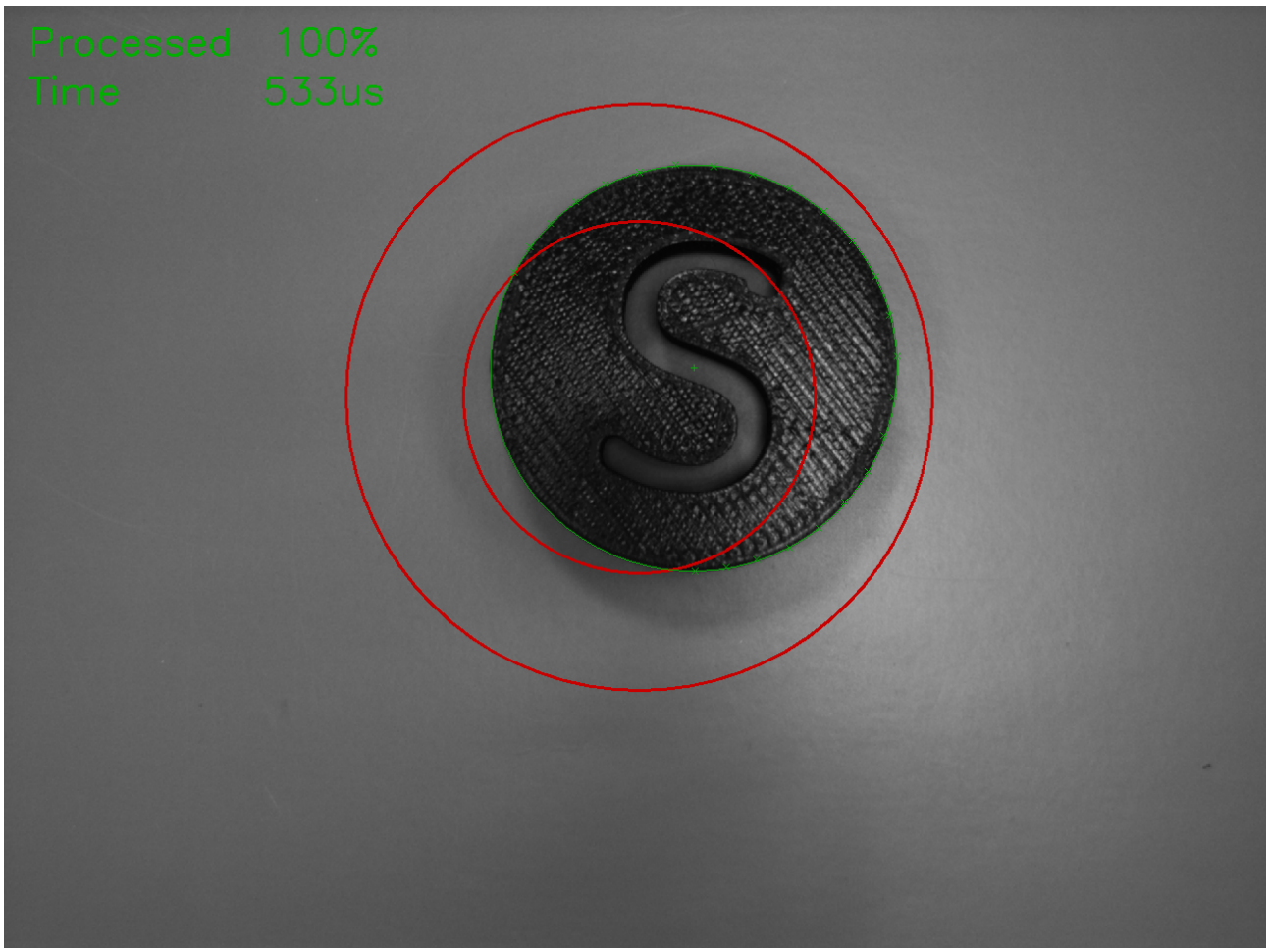


Da die Rechenzeit in der Praxis je nach Eingangsbild etwas schwankt, ist es oftmals sinnvoll, die benötigte Rechenzeit von außen mittels Watchdog zu begrenzen. Setzt man z. B. die Anzahl der Suchlinien auf 120 und den Watchdog timeout `tStop` auf 2000 (abhängig von der verwendeten CPU), so wird die Funktion bei etwa 85% Bearbeitungsanteil abgebrochen und die bis dahin verfügbaren Teilergebnisse zurückgeliefert:



Da der Algorithmus nur an dezidierten Stellen unterbrochen werden und die Teilergebnisse zurückgegeben werden können, bricht die Funktion nicht exakt nach 2000µs ab, sondern benötigt etwas länger, was es bei der Wahl der Abbruchzeit zu berücksichtigen gilt. Die maximal zusätzlich benötigte Zeit ist generell abhängig von dem Algorithmus und der Parametrierung.
Obwohl aufgrund des Abbruchs die Ergebnisse einiger Suchlinien fehlen (4 größere Lücken zwischen den grünen x), konnte die Ellipse trotzdem korrekt approximiert werden, d. h. die Teilergebnisse können sinnvoll weiterverwendet werden, so als wäre die Funktion zu 100% abgearbeitet worden.

In begrenztem Maße ist dies ebenfalls möglich, wenn die Position eines Objekts zum Teil außerhalb des definierten Suchbereichs liegt, z. B. weil ein vorheriger Positionierungsschritt fehlerhaft war. Setzt man z. B. `fMinRadius` auf 180, so liegt bei einigen Beispielbildern ein Teil der Außenkontur außerhalb des vorgegebenen Bereichs. Trotzdem sind noch genügend Suchpunkte vorhanden, um daraus eine Ellipse zu approximieren. In der Praxis sollte man sich allerdings nicht darauf verlassen und den Suchbereich groß genug wählen:



Ähnliche Beispiele

- Locate Circular Arc [▶ 2900]
- Locate Edge [▶ 2904]
- Measure Angle Between Edges [▶ 2914]

8.1.9.4 Measure Angle Between Edges

In diesem Beispiel wird

- mit Hilfe der Funktion F_VN_MeasureAngleBetweenEdgesExp [▶ 1546] der Winkel zwischen zwei Kanten eines Objekts vermessen
- und die Ausführungszeit mit einem Watchdog überwacht und bei Bedarf begrenzt.

Erläuterung

Nach der Festlegung der Suchfenster wurde die Suchrichtung auf `TCVN_ED_LIGHT_TO_DARK` festgelegt, da in diesem Beispiel von innen nach außen gesucht wird und die Objekte heller sind als der Hintergrund.

Der Parameter `nMaxThickness`, der definiert, innerhalb von wie vielen Pixeln `fMinStrength` erreicht werden muss, wurde in diesem Beispiel auf den Wert „`7`“ gesetzt.

Variablen

```
hr              : HRESULT;
hrFunc          : HRESULT;

ipImageIn       : ITcVnImage;
ipImageInDisp   : ITcVnDisplayableImage;
ipImageRes      : ITcVnImage;
ipImageResDisp  : ITcVnDisplayableImage;
```

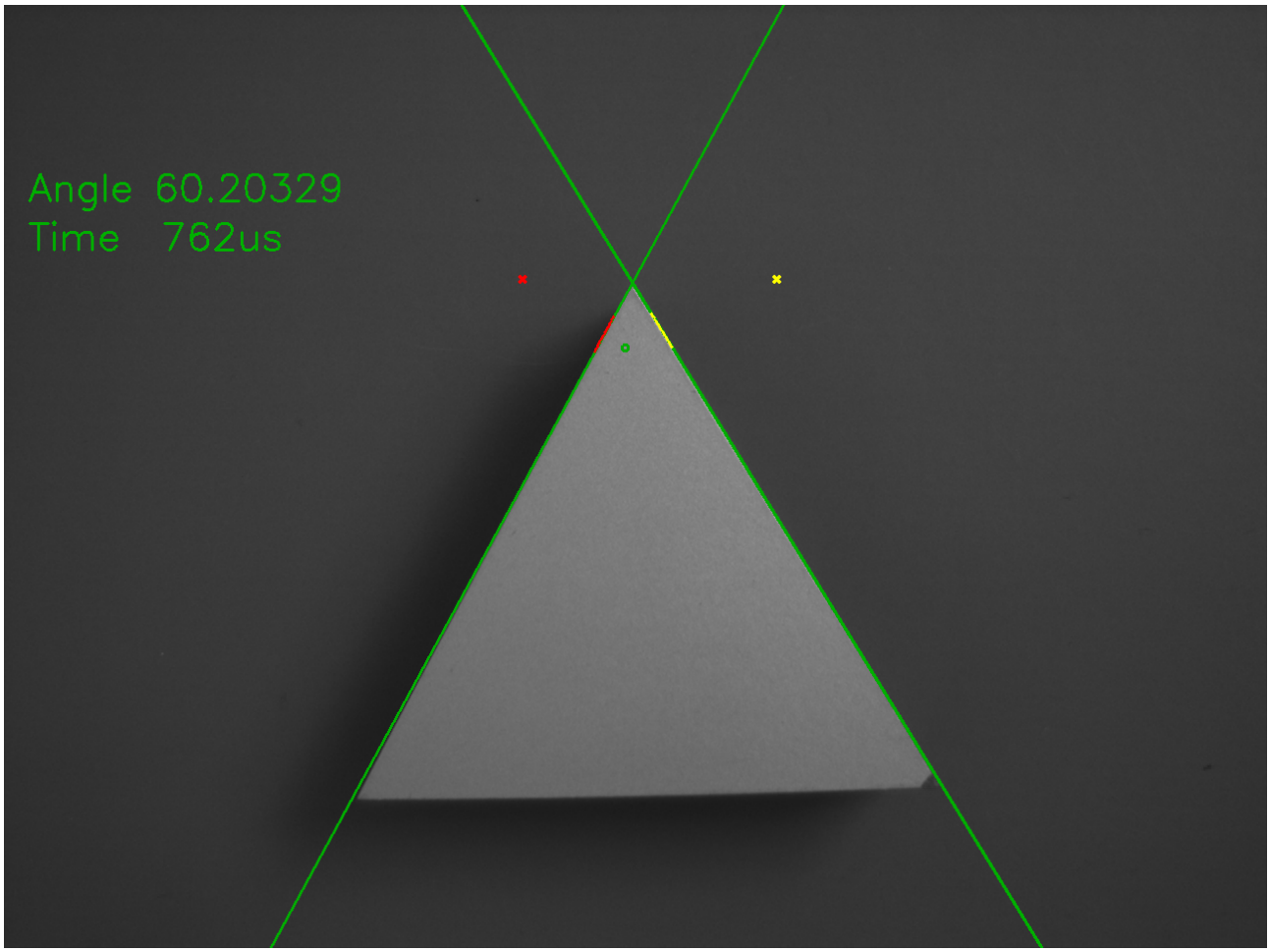
```
// result
fAngle          : REAL;
ipEdgePoints1   : ITcVnContainer;
ipEdgePoints2   : ITcVnContainer;

// parameters
aInnerPoint     : TcVnPoint2_REAL := [635, 350];
aOuterPoint1    : TcVnPoint2_REAL := [530, 280];
aOuterPoint2    : TcVnPoint2_REAL := [790, 280];
fMinStrength    : REAL := 50;
nSearchLines    : UDINT := 41;
fSearchLineDist : REAL := 1;
nSubpixIter     : UDINT := 10;
eAlgorithm      : ETcVnEdgeDetectionAlgorithm := TCVN_EDA_INTERPOLATION;

// Watchdog
hrWD            : HRESULT;
tStop           : DINT := 15000;
tRest           : DINT;
nFraction       : UDINT;

// drawing
aLine1          : TcVnVector4_LREAL;
aLine2          : TcVnVector4_LREAL;
aColorRed       : TcVnVector4_LREAL := [255, 0, 0];
aColorGreen     : TcVnVector4_LREAL := [0, 175, 0];
aColorYellow    : TcVnVector4_LREAL := [255, 255, 0];
sText           : STRING(255);
```

Code

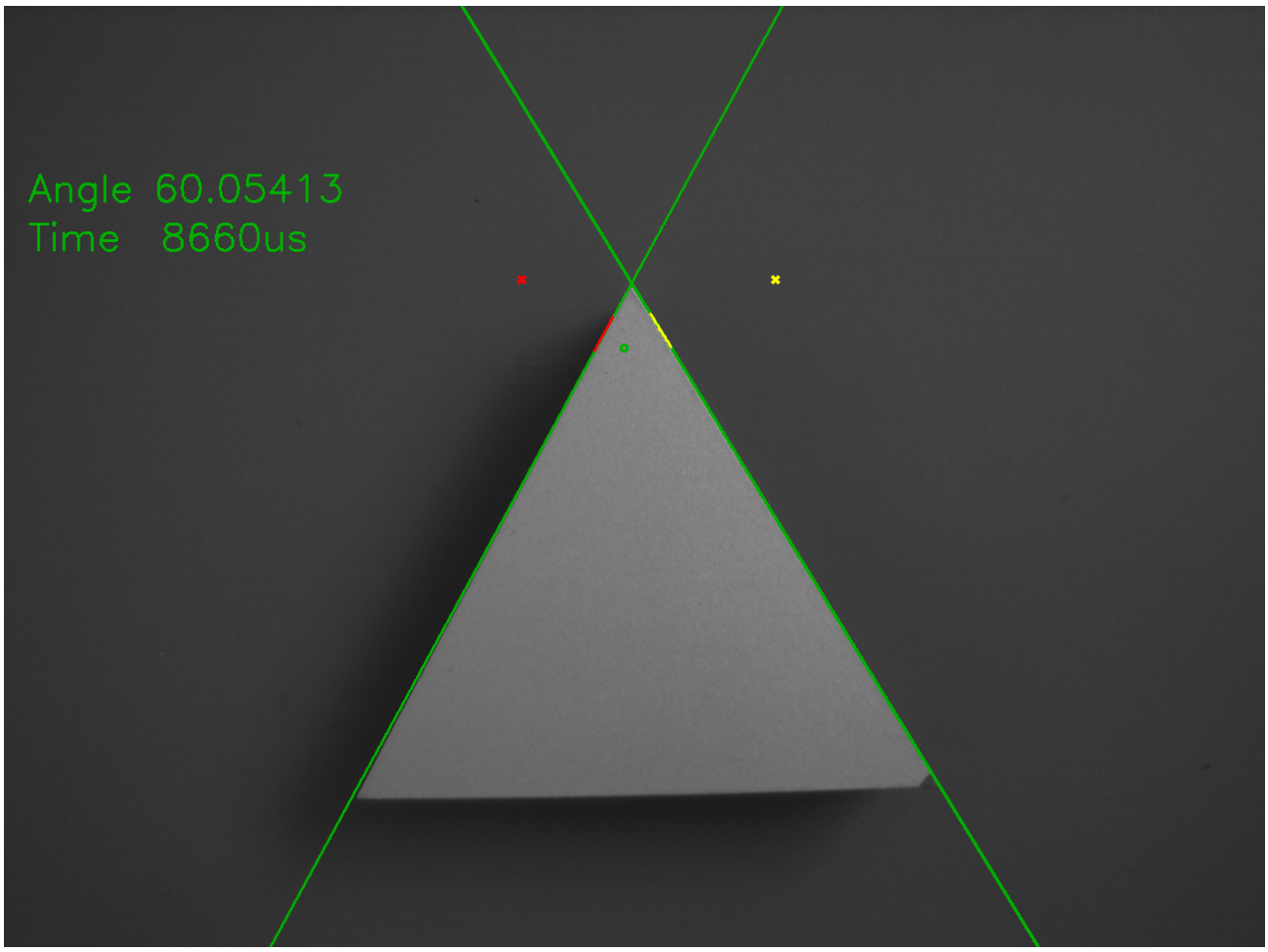
```
hrWD := F_VN_StartRelWatchdog(tStop, hr);
   hrFunc := F_VN_MeasureAngleBetweenEdgesExp(
                ipSrcImage              :=   ipImageIn,
                fAngle                  :=   fAngle
                aInnerPoint             :=   aInnerPoint,
                aOuterPoint1            :=   aOuterPoint1
                aOuterPoint2            :=   aOuterPoint2,
                eEdgeDirection          :=   TCVN_ED_LIGHT_TO_DARK,
                fMinStrength            :=   fMinStrength,
                nSearchLines            :=   nSearchLines,
                fSearchLineDist         :=   fSearchLineDist,
                nMaxThickness           :=   7,
                nSubpixelsIterations    :=   nSubpixIter,
                bAngleInDegrees         :=   TRUE,
                fApproxPrecision        :=   0.0001,
                eAlgorithm              :=   eAlgorithm,
                ipEdgePoints1           :=   ipEdgePoints1,
                ipEdgePoints2           :=   ipEdgePoints2,
                hrPrev                  :=   hr);
hrWD := F_VN_StopWatchdog(hrWD, nFractionProcessed=>nFraction, tRest=>tRest);

// Draw result for visualization
hr := F_VN_ConvertColorSpace(ipImageIn, ipImageRes, TCVN_CST_GRAY_TO_RGB, hr);
sText := CONCAT('Angle ', REAL_TO_STRING(fAngle));
hr := F_VN_PutTextExp(sText, ipImageRes, 25, 200, TCVN_FT_HERSHEY_SIMPLEX, 1.3, aColorGreen, 2,
TCVN_LT_8_CONNECTED, FALSE,hr);
sText := CONCAT(CONCAT('Time ', DINT_TO_STRING(tStop - tRest)), 'us');
hr := F_VN_PutTextExp(sText, ipImageRes, 25, 250, TCVN_FT_HERSHEY_SIMPLEX, 1.3, aColorGreen, 2,
TCVN_LT_8_CONNECTED, FALSE,hr);
hr := F_VN_DrawPointExp(REAL_TO_UDINT(aInnerPoint[0]), REAL_TO_UDINT(aInnerPoint[1]), ipImageRes,
TCVN_DS_CIRCLE, aColorGreen, 3, 2, TCVN_LT_8_CONNECTED, hr);
hr := F_VN_DrawPointExp(REAL_TO_UDINT(aOuterPoint1[0]), REAL_TO_UDINT(aOuterPoint1[1]), ipImageRes,
TCVN_DS_X, aColorRed, 3, 2, TCVN_LT_8_CONNECTED, hr);
hr := F_VN_DrawPointExp(REAL_TO_UDINT(aOuterPoint2[0]), REAL_TO_UDINT(aOuterPoint2[1]), ipImageRes,
TCVN_DS_X, aColorYellow, 3, 2, TCVN_LT_8_CONNECTED, hr);
hr := F_VN_FitLine(ipEdgePoints1, aLine1, hr);
hr := F_VN_FitLine(ipEdgePoints2, aLine2, hr);
hr := F_VN_DrawLine_TcVnVector4_LREAL(aLine1, ipImageRes, aColorGreen, 2, hr);
hr := F_VN_DrawLine_TcVnVector4_LREAL(aLine2, ipImageRes, aColorGreen, 2, hr);
hr := F_VN_DrawPointsExp(ipEdgePoints1, ipImageRes, TCVN_DS_PLUS, aColorRed, 1, 1,
TCVN_LT_8_CONNECTED, hr);
hr := F_VN_DrawPointsExp(ipEdgePoints2, ipImageRes, TCVN_DS_PLUS, aColorYellow, 1, 1,
TCVN_LT_8_CONNECTED, hr);

// Display source and result image
hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, S_OK);
hr := F_VN_TransformIntoDisplayableImage(ipImageRes, ipImageResDisp, S_OK);
```

Ergebnisse

Zur Visualisierung werden zunächst `aInnerPoint` als grüner Kreis, `aOuterPoint1` als rotes x und `aOuterPoint2` als gelbes x dargestellt. Die zugehörigen gefundenen Kantenpunkte `ipEdgePoints1`, `ipEdgePoints2` werden entsprechend ebenfalls in Rot bzw. Gelb eingezeichnet. Die aus den Kantenpunkten approximierten Geraden werden in Grün eingezeichnet. Links im Bild werden der berechnete Winkel `fAngle` in Grad und die benötigte Rechenzeit in µs angezeigt. Für die vorgegebenen Parameter sieht das Ergebnis folgendermaßen aus:



Um ein genaueres Ergebnis zu erhalten, kann `eAlgorithm` auf `TCVN_EDA_APPROX_ERF` geändert werden und passend dazu `nSubpixIter` auf `50`. Allerdings wird dadurch auch die benötigte Rechenzeit deutlich erhöht:



Ähnliche Beispiele

- Locate Circular Arc [▶ 2900]
- Locate Edge [▶ 2904]
- Locate Ellipse [▶ 2909]

8.1.10 Selbstgeschriebene Funktionen

In diesem Beispiel wird erklärt, worauf beim Schreiben von eigenen Funktionen für Vision-Applikationen zu achten ist. Dabei geht es im Wesentlichen um:

- Handhabung des HRESULT [▶ 150]
- Handhabung von Interface Pointern [▶ 146]

Auch andere Strukturen

Diese Funktionsimplementierung ist an die Struktur der TwinCAT Vision API-Funktionen angelehnt. Es sind auch andere Strukturen denkbar, achten Sie jedoch auf die hier erklärten Punkte.

Auch Methoden

In diesem Beispiel wird eine Funktion beschrieben. Es ist jedoch in gleichem Maße für Methoden gültig.

Dazu wird beispielhaft eine selbst geschriebene Funktion zum Zählen und Malen von Objekten betrachtet:

```
FUNCTION F_CountAndDrawObjects : HRESULT
```

Deklarationsteil

Die Attribute der Funktion sind wie folgt im `VAR_INPUT`-Teil deklariert. Das Ursprungsbild wird direkt als `ITcVnImage` übergeben. Da in einer Funktion oder Methode keine Variablenwerte gespeichert werden, muss in diesem Fall nichts weiter beachtet werden. Hingegen wird das Ergebnisbild als `REFERENCE TO ITcVnDisplayableImage` übergeben, da es in der Funktion erstellt wird und nach außen zur weiteren Verarbeitung zurückgegeben werden soll. Hier wird aufgrund der Reference also keine Kopie des Interface Pointers erstellt.

```
VAR_INPUT
    ipSrcImage          :   ITcVnImage;
    ipDestImage         :   REFERENCE TO ITcVnImage;
    aColor              :   TcVnVector4_LREAL;
    nNumberOfObjects    :   REFERENCE TO ULINT;
    hrPrev              :   HRESULT;
END_VAR
```

Zudem werden einige Hilfsvariablen im `VAR` deklariert, die intern von der Funktion benötigt werden.

```
VAR
    hr                  :   HRESULT;
    ipContours          :   ITcVnContainer;
    stParams            :   TcVnParamsBlobDetection;
END_VAR
```

Eingangsprüfungen

Am Anfang der Funktion werden einige Eingangsprüfungen durchgeführt, um sicherzustellen, dass die Funktion ordnungsgemäß ausgeführt werden kann.

Zunächst wird mittels `FAILED(hrPrev)` überprüft, ob in der bisherigen Verarbeitungskette ein Fehler aufgetreten ist. Wenn dies der Fall ist, wird derselbe Fehler zurückgegeben und die Funktion direkt beendet.

```
IF FAILED(hrPrev) THEN
    F_CountAndDrawObjects := hrPrev;
    RETURN;
END_IF
```

Ebenfalls wird überprüft, ob alle Interface Pointer, die als VAR_INPUT an die Funktion übergeben werden, valide sind. Anderenfalls kann die Funktion nämlich nichts damit anfangen und gibt als Return Code INVALIDPARM zurück.

```
IF ipSrcImage = 0 THEN
    F_CountAndDrawObjects := Tc2_System.E_HRESULTAdsErr.INVALIDPARM;
    RETURN;
END_IF
```

Sofern alle Eingangsprüfungen erfolgreich waren, müssen z. B. bei Funktionsblöcken noch die Referenzzähler aller als `VAR_INPUT` übergebenen Interface Pointer inkrementiert werden. Denn durch das Übergeben des Interface Pointers wird eine Kopie erstellt, die nach dem Ausführen des Bausteins erhalten bleiben und daher im Referenzzähler abgebildet werden müssen. Die Methode `TcAddRef` kann direkt ohne vorherige `<> 0` Prüfung ausgeführt werden, da dies schon während der Eingangsprüfungen geschehen ist. Wie schon im Deklarationsteil beschrieben, ist das Erhöhen des Referenzzählers bei Funktionen und Metthoden nicht notwendig, da die Variablenwerte wie in diesem Fall die Pointer-Adresse nach der Ausführung nicht erhalten bleiben.

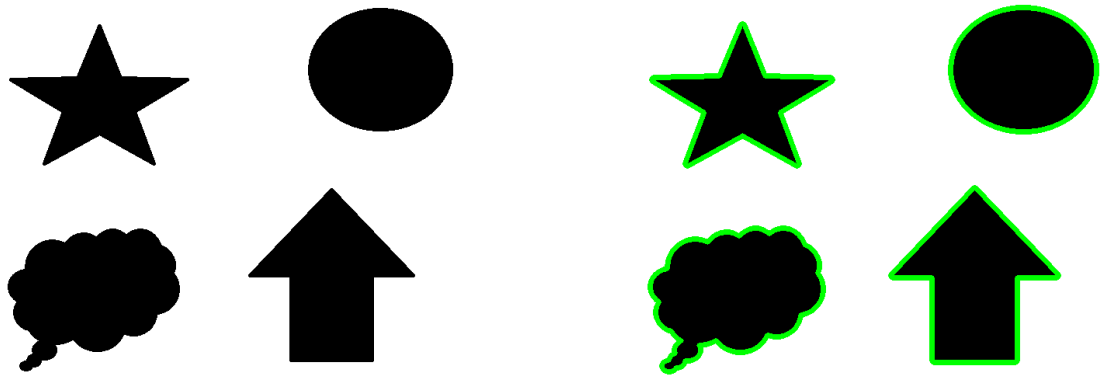
Hauptverarbeitung

Im Hauptteil der Funktion steht der tatsächliche Bildverarbeitungsablauf. Dieser ist hier nur beispielhaft und wird nicht weiter betrachtet. Der Aufruf von `F_VN_ConvertColorSpace` verdeutlicht, dass Interface Pointer, die zum Zwecke der Ergebnisrückgabe als `REFERENCE TO` übergeben werden, direkt von TwinCAT Vision API-Funktionen beschrieben werden können.

```
hr := F_VN_ConvertColorSpace(ipSrcImage, ipDestImage, TCVN_CST_GRAY_TO_RGB, hr);
stParams.bFilterByArea := TRUE;
stParams.fMinArea := 10_000;
stParams.fMaxArea := 100_000;
hr := F_VN_DetectBlobs(ipSrcImage, ipContours, stParams, hr);
hr := F_VN_GetNumberOfElements(ipContours, nNumberOfObjects, hr);
hr := F_VN_DrawContours(ipContours, -1, ipDestImage, aColor, 5, hr);
```

Durch diesen Bildverarbeitungsablauf werden Objekte im Ergebnisbild gemalt und ihre Anzahl als ULINT zurückgegeben:

Ursprungsbild **Ergebnisbild**



Anzahl gefundener Objekte: 4

Aufräumen und Rückgabe

Nach dem eigentlichen Bildverarbeitungsablauf müssen die Interface Pointer, die innerhalb der Funktion deklariert sind, wieder freigegeben werden. Dies ist der Fall, weil alle in der Funktion deklarierten Variablen (also auch die Interface Pointer) nach Beenden der Funktion gelöscht werden. Wenn im Hintergrund nun noch Daten vorhanden wären, die noch nicht freigegeben sind, entständen Speicher-Lecks.

```
FW_SafeRelease(ADR(ipContours));
```

Schließlich wird das `HRESULT` als Rückgabewert der Funktion zugewiesen, um eventuelle Fehler nach außen zu melden. Damit es nicht verfälscht wird, sollten die Rückgabewerte der `FW_SafeRelease` Funktionen nicht dem `HRESULT` zugewiesen werden.

```
F_CountAndDrawObjects := hr;
```

8.2 Beispiele zu Funktionsblöcken

In diesem Abschnitt finden Sie Beispiele zu den Function Blocks [▶ 1620].

8.2.1 Camera Register Access

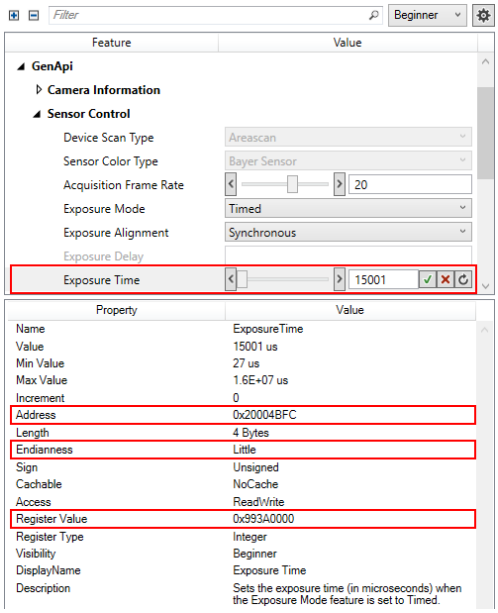
8.2.1.1 Lesen/Schreiben eines Registers

Alle Kamera-Parameter die über den Konfigurationsassistanten im Config Mode gelesen bzw. geschrieben werden können, können auch aus der SPS über die Funktionsbausteine FB_VN_ReadRegister_UDINT [▶ 1626] und FB_VN_ReadRegister_REAL [▶ 1624] gelesen bzw. über FB_VN_WriteRegister_UDINT [▶ 1634] und FB_VN_WriteRegister_REAL [▶ 1632] geschrieben werden. Welcher Baustein benötigt wird, hängt vom Datentyp des Parameters ab und kann in der Property-Auflistung am Register Type abgelesen werden. In diesem Beispiel wird die Belichtungszeit geschrieben und gelesen, wobei es sich um eine Ganzzahl handelt.

```
fbReadValue    :    FB_VN_ReadRegister_UDINT;
fbWriteValue   :    FB_VN_WriteRegister_UDINT;
```

Beide Bausteine benötigen die Adresse des Parameters und eine Angabe zur Byte-Reihenfolge (Endianness). Der Write-Baustein benötigt zusätzlich noch den Parameterwert, der an die Kamera geschrieben werden soll. Da der unter Value angezeigte oder eingegebene Wert nicht direkt mit dem übereinstimmen muss wie ihn die Kamera intern verarbeitet, z. B. bei Enums oder anderen Byte-Reihenfolgen, kann dieser unter dem Register **Value** ausgelesen werden.

Diese Informationen können aus dem Konfigurationsassistenten ausgelesen werden:



HINWEIS

Kameraspezifische Parameter

Die hier abgebildeten Parameternamen, Properties und Werte sind Hersteller- bzw. Kamera- und Firmware-spezifisch und können daher abweichen.

Variablen

```
bTriggerReadValue  :   BOOL;
nReadValue         :   UDINT;
fbReadValue        :   FB_VN_ReadRegister_UDINT;
nReturnCodeRead    :   UDINT;

bTriggerWriteValue :   BOOL;
nWriteValue        :   UDINT := 5000;
fbWriteValue       :   FB_VN_WriteRegister_UDINT;
nReturnCodeWrite   :   UDINT;
```

Code

```
fbReadValue(
    nAddress    :=   16#20004BFC, // Check the Address of the Camera Parameter
    nEndian     :=   1,           // 0 = Big, 1 = Little
    bRead       :=   bTriggerReadValue,
    nTimeout    :=   T#5S,
    nValue      =>   nReadValue);

fbWriteValue(
    nAddress    :=   16#20004BFC, // Check the Address of the Camera Parameter
    nValue      :=   nWriteValue, // Check the right Input Format
    nEndian     :=   1,           // 0 = Big, 1 = Little
    bWrite      :=   bTriggerWriteValue,
    nTimeout    :=   T#5S);
```

Bedingungen zum Lesen und Schreiben von Parametern

Parameter können generell nur bei geöffneter Kameraverbindung (`ETcVnCameraState >= TCVN_CS_OPENED`) gelesen oder geschrieben werden. Die genauen Bedingungen entnehmen Sie der entsprechenden Herstellerdokumentation der Kamera.

Beide Bausteine müssen mit dem **Image Provider** der Kamera initialisiert werden:

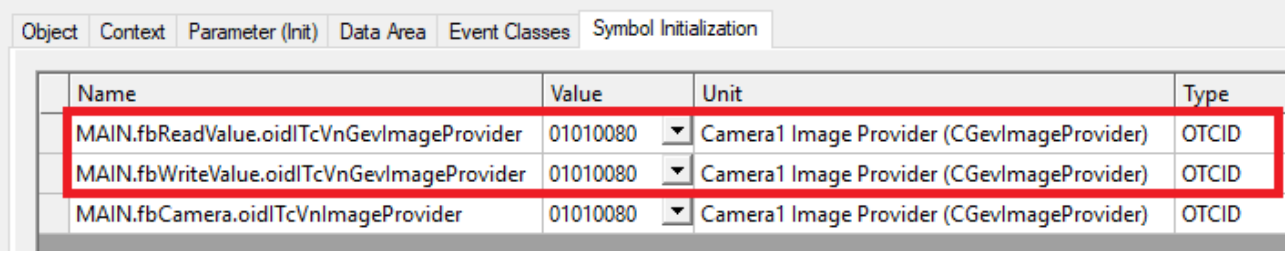


Abb. 31: Verknüpfung von Funktionsblöcken mit einer Kamera-Instanz

8.2.2 File Access

8.2.2.1 Speichern von Bildern aus der SPS

In diesem Beispiel werden Bilder aus der SPS auf einem lokalen Laufwerk gespeichert. Dazu wird der Funktionsbaustein FB_VN_WriteImage [▶ 1656] verwendet.

Der Speichervorgang erfolgt asynchron über mehrere Zyklen.

Dauer des Speichervorgangs

Die Dauer des Speichervorgangs hängt von verschiedenen Faktoren ab und kann daher nicht pauschal angegeben werden.

Variablen

```
hr                    :   HRESULT;

fbCamera              :   FB_VN_SimpleCameraControl;

ipImageIn             :   ITcVnImage;
ipImageInDisp         :   ITcVnDisplayableImage;

// Sample Specific Variables
fbWriteImage          :   FB_VN_WriteImage := (nTimeout := T#500MS);
sFilePath             :   STRING(255) := '';
bWriteImageTrigger    :   BOOL;
bWriteImageWaitResult:    BOOL;
bWriteImageDone       :   BOOL;
nReturnCode           :   UDINT;
```

In diesem Beispiel kann ein Bild gespeichert werden, indem die Trigger-Variable `bWriteImageTrigger` auf `TRUE` gesetzt wird.

Code

Das übergebene `ipImage` wird bereits beim ersten Aufruf des `fbWriteImage` mit steigender Flanke an `bWrite` komplett übernommen und kann daher direkt im Anschluss z. B. mit `F_VN_TransformIntoDisplayableImage` freigegeben werden. Danach muss der `fbWriteImage` weiterhin zyklisch aufgerufen werden, bis der `bBusy` Ausgang wieder `FALSE` ist. Dies kann bei einem Fehler auch schon beim ersten Aufruf der Fall sein, wenn z. B. gar keine oder eine falsche Dateiendung angegeben wird. Eine Auswertung, ob das Schreiben erfolgreich war oder ob ein Fehler aufgetreten ist, wird im unteren IF-Block durchgeführt. Im Erfolgsfall wird das `bWriteImageDone` gesetzt und bleibt, bis zum nächsten Setzen von `bWriteImageTrigger` anstehen.

```
IF SUCCEEDED(hr) AND ipImageIn <> 0 THEN

    IF NOT fbWriteImage.bBusy AND bWriteImageTrigger THEN
        bWriteImageTrigger := FALSE;
        bWriteImageDone := FALSE;
        bWriteImageWaitResult := TRUE;

        // With setting sFilePath:= '' to an empty string the images are saved under the default
```

```
path
        // The default path can be set at the service configuration tab of the Vision Node
        // If sFilePath is set, it must contain the full path, image name and type
        // for example: sFilePath := 'C:\WriteImage\ImageName.bmp'
        fbWriteImage(ipImage := ipImageIn, sFilePath := sFilePath, bWrite := TRUE);
    END_IF

    hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, hr);
END_IF

fbWriteImage(sFilepath:= '', bWrite := FALSE);

IF bWriteImageWaitResult AND NOT fbWriteImage.bBusy THEN
    bWriteImageWaitResult := FALSE;
    // Check if fbWriteImage was finished successfully or implement error handling
    IF fbWriteImage.bError THEN
        nReturnCode := fbWriteImage.nErrorId AND 16#FFF;
    ELSE
        bWriteImageDone := TRUE;
        nReturnCode := 0;
    END_IF
END_IF
```

8.2.2.2 Serielles Laden von Bildern in die SPS

In diesem Beispiel laden Sie Bilder seriell in die SPS. Dazu nutzen Sie:

- FB_VN_ReadImage [▶ 1648]

Dieses Beispiel gilt analog auch für das Einladen von Containern mit FB_VN_ReadContainer [▶ 1646].

Erklärung

Ein häufiger Anwendungsfall ist, dass Referenzbilder zum Vergleich mit dem Live-Bild in der Steuerung hinterlegt sind. Diese Referenzbilder sollen im laufenden Betrieb ausgewechselt werden, z. B. wenn eine neue Referenz eingeführt wird. Das Nachladen von Bildern darf beliebig oft und in beliebigen Abständen geschehen.

Um nicht für jedes neue Referenzbild zunächst den Dateinamen der SPS mitteilen zu müssen, legen Sie eine Namenskonvention fest. Diese kann z. B. `Image<Index>.png` sein, wobei `<Index>` durch einen fortlaufenden Integer zu ersetzen ist. Jedes neue Referenzbild erhält dann einen Dateinamen mit einem um 1 inkrementierten Index. Somit weiß die SPS wie das nächste Referenzbild heißen muss und kann dieses gezielt versuchen einzuladen.

HINWEIS

Dateikorruption droht

Ersetzen Sie nie den Inhalt einer Bilddatei, sondern legen Sie nur neue Bilder im Dateisystem an! Zum Einen werden ersetzte Bilder aufgrund von internen Caching-Mechanismen vom Vision-Service nicht registriert. Zum Anderen kann der gleichzeitige Zugriff mehrerer Prozesse auf das eine Bild zu einer Dateikorruption führen. Die Cache-Einstellungen können alternativ angepasst werden.

Anwendung

Diesem Beispiel liegen 6 farbige Bilder mit der Benennung `image<Index>.png` sowie der leere Ordner `load` bei. Legen Sie den absoluten Pfad des `load`-Ordners als `sBasePath` fest. Nachdem Sie die Beispiel-SPS starten, verschieben Sie der Reihenfolge nach (angefangen bei `Index=0`) die Bilder in den `load`-Ordner und beobachten im ADS Image Watch, dass sich entsprechend das Bild `ipImageRefDisp` verändert.

Programm

Im zyklischen Programm errechnen Sie zunächst den Pfad der einzuladenden Datei, basierend auf dem Index `nIndex`. Passen Sie zudem den Basispfad `sBasePath` in der Variablendeklaration an, sodass er mit dem Speicherort Ihrer Bilder übereinstimmt.

```
sPath := CONCAT(CONCAT(sBasePath, TO_STRING(nIndex)), '.png');
```

Mit dem errechneten Pfad rufen Sie den Funktionsblock fbReadImage vom Typ FB_VN_ReadImage [▶ 1648] zyklisch auf.

```
fbReadImage(
    sFilePath   :=  sPath,
    ipDestImage :=  ipImageIn,
    bRead       :=  TRUE,
    nTimeout    :=  T#1S
);
```

Neben dem Aufruf des Funktionsblocks müssen Sie reagieren, wenn dieser fertig ist. Zum einen setzen Sie den Funktionblocks durch einen Aufruf mit `bRead:=FALSE` zurück; zum anderen erhöhen Sie den Index und geben das Bild an den gewünschten Interface Pointer weiter, sofern die Ausführung erfolgreich ist. Die Abfrage auf einen Fehler ist sehr wichtig, da Sie den Index nur erhöhen dürfen, wenn das aktuelle Bild bereits erfolgreich geladen wurde.

```
IF NOT fbReadImage.bBusy THEN
    IF NOT fbReadImage.bError THEN
        nReturnCode := 0;
        nIndex := nIndex + 1;

        IF ipImageIn <> 0 THEN
            FW_SafeRelease(ADR(ipImageRef));
            ipImageRef := ipImageIn;
            ipImageIn.TcAddRef();
            FW_SafeRelease(ADR(ipImageIn));
        END_IF
    ELSE
        nReturnCode := fbReadImage.nErrorId AND 16#FFF;
    END_IF
    fbReadImage(sFilePath:='', bRead:=FALSE);
END_IF
```

Die Weitergabe des Bildes geschieht hier über eine ordentliche Übergabe des Interfaces und nicht über eine Kopie, um Speicher und Rechenzeit zu sparen.

8.2.3 Image Acquisition

8.2.3.1 Bild per Dateiname triggern

In diesem Beispiel wird ein Bild gezielt anhand des Dateinamens im Funktionsblock FB_VN_FileSourceControl [▶ 1664] angetriggert.

Anwendung

Das Beispiel zeigt abwechselnd zwei Bilder mit den Dateinamen *Image1.bmp* und *Image2.bmp* aus dem File Source im ADS Image Watch [▶ 141] an. Die Bilder mit entsprechenden Dateinamen liegen dem Beispiel bei. Sie können die Beispielbilder durch eigene Bilder ersetzen oder diese zusätzlich hinzufügen. Ändern oder ergänzen Sie dazu die Dateinamen in `sFileName` und passen Sie ggf. die Anzahl der Elemente an. Achten Sie darauf, dass zuerst alle anzuzeigenden Bilder im File Source Control [▶ 136] hinzufügt werden müssen. Durch Setzen von `bLoop` auf `false` stoppen Sie den automatischen Wechsel der Bilder und können dann durch Setzen von `nImageIndex` das Bild auswählen, welches getriggert werden soll.

Programm

```
hr              :   HRESULT;
sFileName       :   ARRAY [0..1] OF STRING := ['Image1.bmp', 'Image2.bmp'];
fbFileSource    :   FB_VN_FileSourceControl;
eState          :   ETcVnCameraState;
ipImageIn       :   ITcVnImage;
ipImageInDisp   :   ITcVnDisplayableImage;
bTrigger        :   BOOL := TRUE;
bLoop           :   BOOL := TRUE;
nImageIndex     :   UINT := 0;
```

fbFileSource	Funktionsblock zur Steuerung des File Source [▶ 135] vom Typ FB_VN_FileSourceControl [▶ 1664].
eState	Zustand des File Source
ipImageIn	Eingangsbild
ipImageInDisp	Anzeigbares Eingangsbild
bTrigger	Trigger-Flag
sFileName	Dateiname - ändern Sie diese Variable, um ein anderes Bild einzuladen. Dieses Bild muss allerdings im File Source Control [▶ 136] hinzugefügt werden.
hr	Statusvariable vom Typ HRESULT [▶ 150].

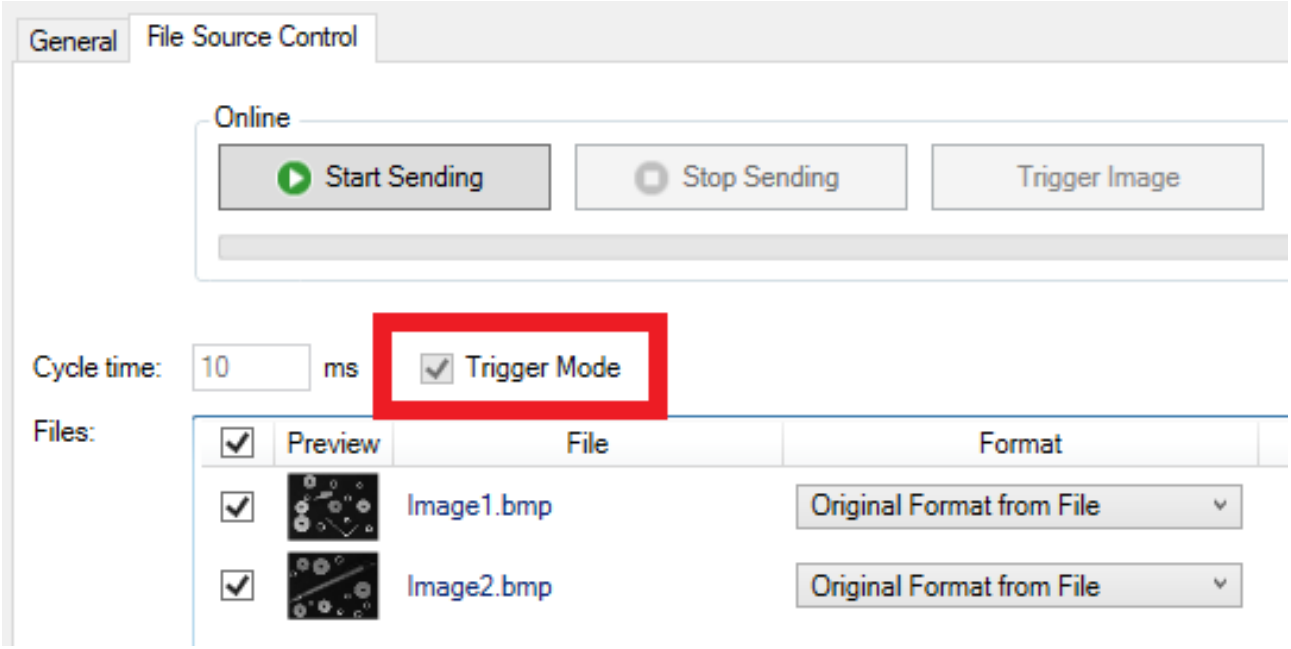
Grundsätzlich wird der übliche Zustandsautomat genutzt, um das File Source zu bedienen:

```
eState := fbFileSource.GetState();
CASE eState OF

<...>

END_CASE
```

Das Zurücksetzen bei Fehlern und das Starten der Bildaufnahme geschieht ganz normal in den entsprechenden Zuständen:

```
TCVN_CS_INITIAL, TCVN_CS_INITIALIZING, TCVN_CS_INITIALIZED, TCVN_CS_OPENING, TCVN_CS_OPENED,
TCVN_CS_STARTACQUISITION:
    fbFileSource.StartAcquisition();

TCVN_CS_ERROR:
    fbFileSource.Reset();
```

Damit der File Source allerdings nicht sofort Bilder einlädt, wenn die Bildaufnahme gestartet wird, muss der **Trigger Mode** im File Source Control aktiviert werden:



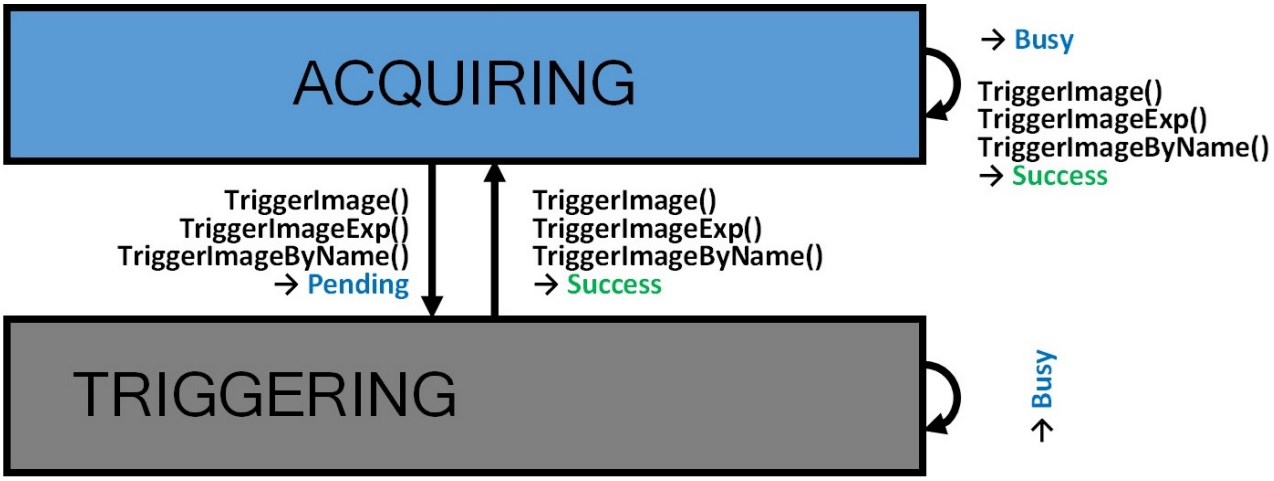
Im ACQUIRING Zustand wird nun abwechselnd das Bild mit entsprechendem Dateinamen getriggert und daraufhin empfangen. Zum Triggern des Bildes wird hier explizit die Methode TriggerImageByName [▶ 1672] verwendet. Diese Methode existiert nur im Funktionsblock FB_VN_FileSourceControl [▶ 1664] und nicht in FB_VN_SimpleCameraControl [▶ 1697].

```
TCVN_CS_ACQUIRING:
    IF bTrigger THEN
        hr := fbFileSource.TriggerImageByName(sFileName[nImageIndex]);
        IF SUCCEEDED(hr) THEN
            bTrigger := FALSE;
```

```
            IF bLoop THEN
                nImageIndex := nImageIndex + 1;
                IF nImageIndex >= 2 THEN
                    nImageIndex := 0;
                END_IF
            END_IF
        END_IF
    ELSE
        hr := fbFileSource.GetCurrentImage(ipImageIn);
        IF SUCCEEDED(hr) AND ipImageIn <> 0 THEN
            bTrigger := TRUE;
            hr := F_VN_TransformIntoDisplayableImage(ipImageIn, ipImageInDisp, hr);
        END_IF
    END_IF
```

Da der File Source beim Triggern des Bildes zwischenzeitlich in den `TRIGGERING` Zustand geht, muss in diesem Zustand weiterhin getriggert werden. Dies geschieht so lange, bis der File Source das Bild vollständig eingeladen hat und sein Zustand somit wieder auf `ACQUIRING` wechselt. Beim weiteren Triggern muss nicht erneut der Dateiname des Bildes übergeben werden; es reicht ein Aufruf der Methode TriggerImage [▶ 1671].

```
TCVN_CS_TRIGGERING:
    fbFileSource.TriggerImage();
```



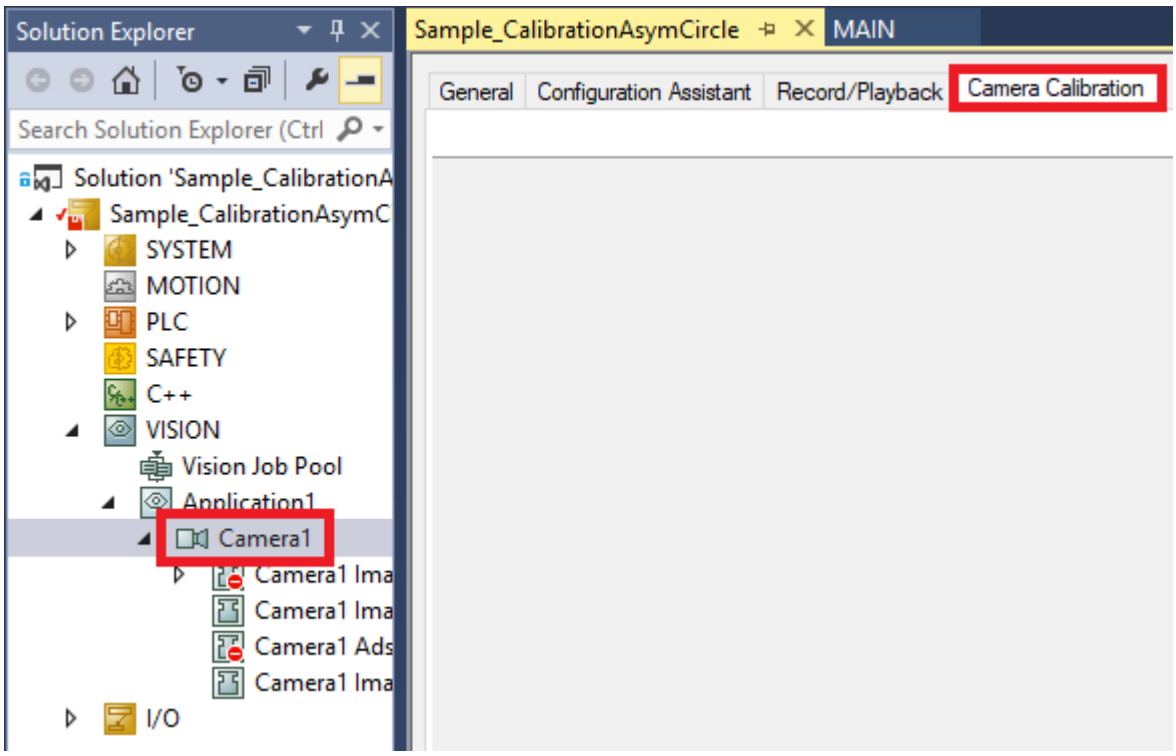
Die Variable `bTrigger` wird in diesem Beispiel aus Demonstrationszwecken automatisch auf `TRUE` gesetzt, sobald ein Bild erfolgreich geladen wurde. In einer echten Applikation würde der Trigger an einer ablauftechnisch sinnvollen Stelle gesetzt werden.

8.3 Beispiele zu Assistenten

8.3.1 Calibration Assistent

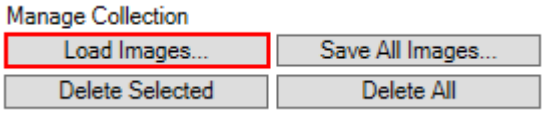
Zur Umrechnung von Bildpunkten in Weltpunkte ist eine geometrische Kamera-Kalibrierung erforderlich, bei der die dafür benötigten Parameter bestimmt werden. In diesem Beispiel wird die geometrische Kalibrierung einer Kamera mit Hilfe eines asymmetrischen Kreismusters und des Kalibrierungsassistenten gezeigt. Die Schritte können analog auch für die anderen Kalibriermuster (siehe Camera Calibration [▶ 116]) durchgeführt werden. Lediglich Schritt 2 muss entsprechend angepasst werden.

Um dieses Beispiel durchzuführen, öffnen Sie zuerst das Projekt mit angelegter Kamera und danach den Kalibrierungsassistenten der Kamera-Instanz.



Schritt 1 – Bild-Akquise

Als erstes müssen Bilder des Kalibrationsmusters in den Assistenten geladen werden, die von der zu kalibrierenden Kamera gemacht wurden. Dies kann entweder über eine direkte Bildaufnahme mit der Kamera oder durch das Einladen von existierenden Bildern geschehen. Für dieses Beispiel finden Sie im zugehörigen Bildordner vier geeignete Bilder *ImgCalibAsymCirclesx.bmp* eines asymmetrischen Kreismusters. Laden Sie diese Bilder über **Load Images...** in den Kalibrierungsassistenten.

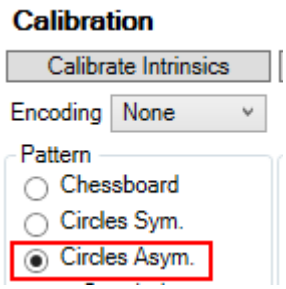


Schritt 2 – Asymmetrisches Kreismuster definieren

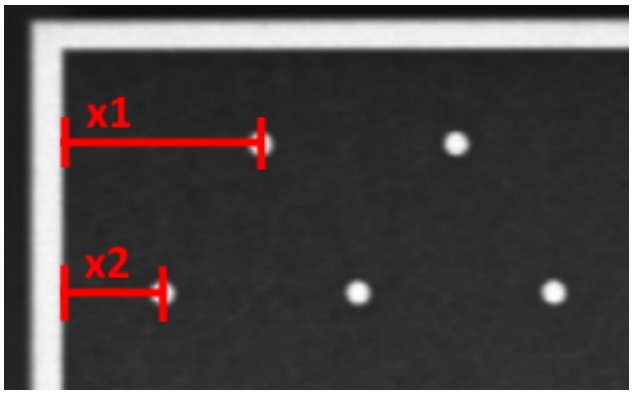
In diesem Beispiel wird als Kalibriermuster ein asymmetrisches Kreismuster genutzt. Um einen der anderen Mustertypen zu verwenden, beachten Sie die jeweilige Definition:

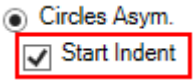
- Schachbrettmuster [▶ 126]
- Symmetrische Kreismuster [▶ 127]
- Asymmetrische Kreismuster [▶ 128]
- Individuelle Kreismuster [▶ 129]

Für die eingeladenen Beispielbilder muss das asymmetrische Kreismuster im Kalibrierungsassistenten angewählt werden:

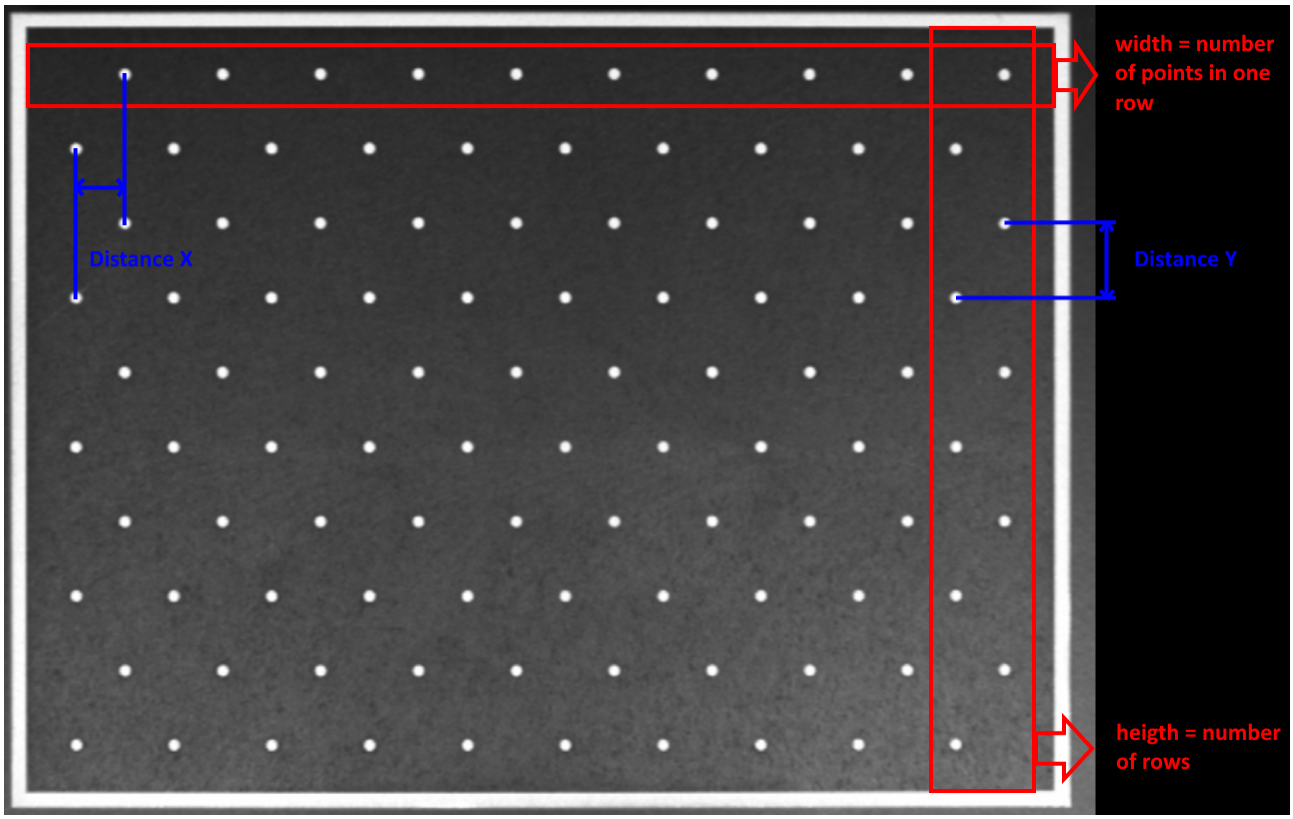


Bedingt durch die Asymmetrie ist entweder die erste (x1) oder die zweite (x2) Zeile des Kreismusters eingerückt. Da in diesem Fall die erste (x1) Zeile eingerückt ist, wählen Sie **Start Indent**.

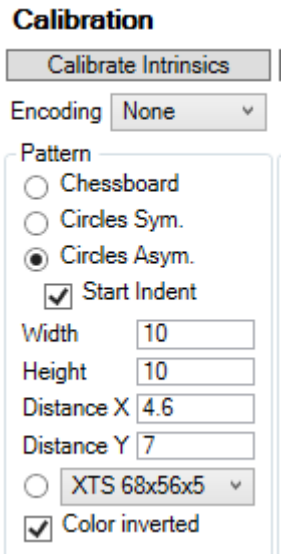




Die Breite des Musters **Width** entspricht der Anzahl von Punkten/Kreisen in einer Zeile, während die Höhe **Height** der Zeilenanzahl entspricht. Bei **Distance X** wird der kürzeste Abstand in X-Richtung zwischen zwei Kreisen angegeben. Dies ist beim asymmetrischen Muster der Abstand zum nächsten Kreis in der darunter / darüber liegenden Zeile, Einzeichnung im Bild beachten. Der Abstand zwischen zwei Zeilen wird in **Distance Y** angegeben.

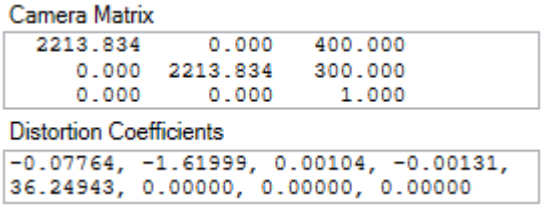


Der Assistent erwartet schwarze Kreise auf weißen Grund. Da in diesem Beispiel ein Muster bestehend aus weißen Kreisen auf schwarzen Grund verwendet wird, muss die Farbe invertiert werden, wählen Sie dazu **Color inverted** an.



Schritt 3 – Intrinsische Parameter berechnen

Nach der vollständigen Definition des Kalibriermusters können die intrinsischen Parameter über den Button **Calibrate Intrinics** berechnet werden. Anschließend sollte der **Reproj. Error** zwischen 0 und 1 liegen und die **Kamera Matrix**, sowie die **Distortion Coefficients** bestimmt sein.

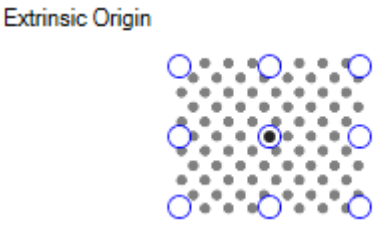


Ist dies nicht der Fall, gibt der Tooltip beim **Reproj. Error** Hinweise auf den möglichen Fehler. Zudem wird unter *C:\ProgramData\Beckhoff\Vision_CalibrationAssistantOutput* eine Bildserie erstellt, die weiteren Aufschluss auf die Erkennung des Musters gibt. Schreibrechte auf das Verzeichnis sind notwendig.

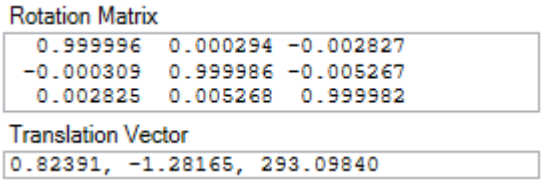
Schritt 4 – Extrinsische Parameter berechnen

Nach den intrinsischen Parametern können die extrinsischen Parameter berechnet werden. Dazu wird nur noch ein Bild verwendet, und zwar das Bild, bei dem sich das Muster in der Ebene befindet, auf der zum Beispiel später die Bauteile vermessen werden. Von den Beispielbildern ist dies das Erste. Klicken Sie daher das erste eingeladene Bild an, sodass es in der Bildvorschau erscheint.

Zudem kann der Nullpunkt des Musters **Extrinsic Orgin** angegeben werden. Standardmäßig ist dieser in der Mitte des Musters.

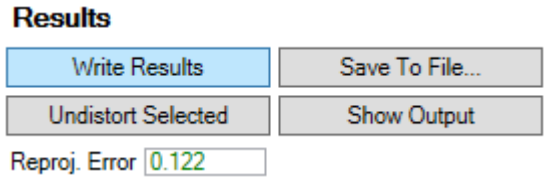


Über den Button **Calibrate Extrinsics** werden anschließend die **Rotation Matrix** und der **Translation Vector** berechnet. Die Ergebnisse finden Sie wieder unter **Results**.

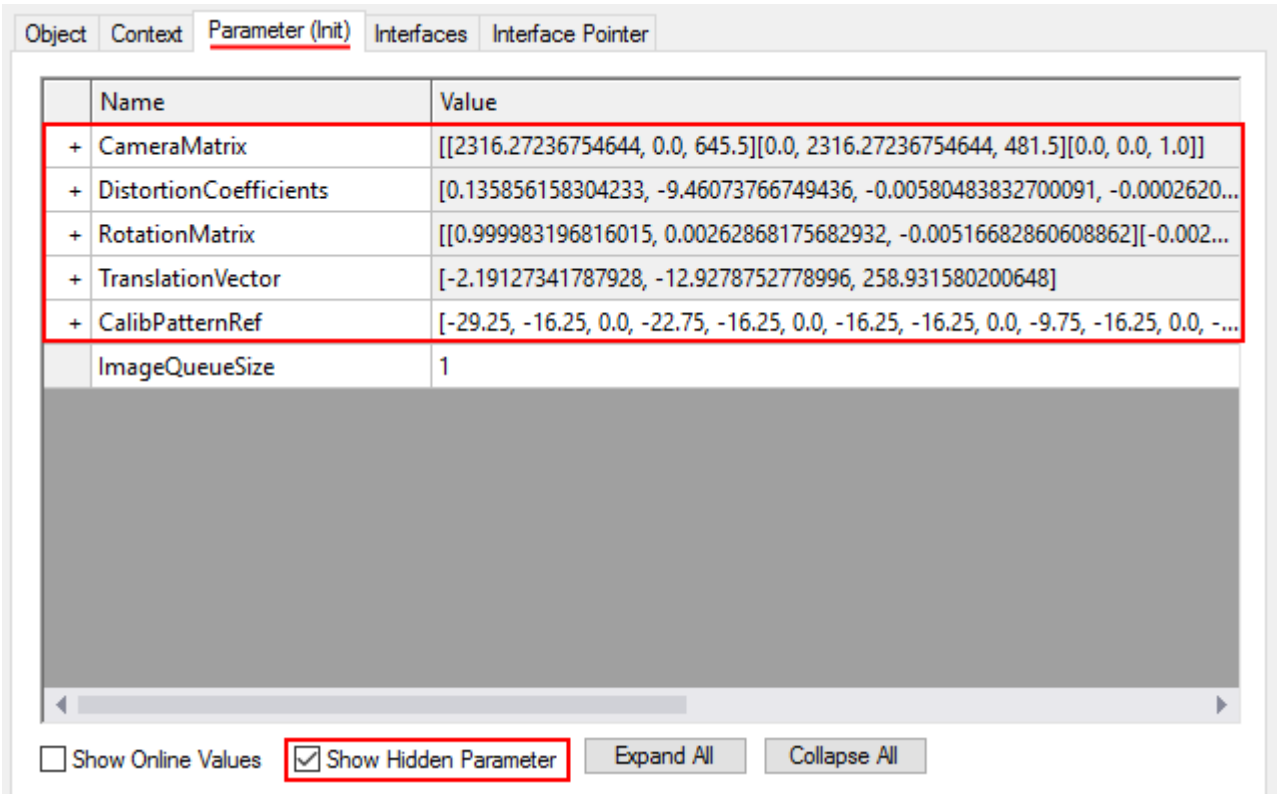


Schritt 5 – Ergebnisse in den Image Provider der Kamera schreiben

War die Kalibrierung erfolgreich, können Sie die Ergebnisse über **Write Results** in den Image Provider der Kamera schreiben. Dieser Schritt ist nötig, damit der Kamera-Funktionsbaustein FB_VN_GevCameraControl [▶ 1672] in der SPS auf die Kalibrationsergebnisse zugreifen kann.



Im Image Provider sind die Kalibrierungsergebnisse unter **Parameter (Init)** zu finden, wenn **Show Hidden Parameter** ausgewählt ist.



Achten Sie darauf, nach diesem Schritt durch einen Klick auf die Konfiguration zu aktivieren. Anderenfalls stehen die Kalibrationsergebnisse nicht in der SPS zur Verfügung.

Schritt 6 – Kalibrationsergebnisse in die SPS laden

Über den Funktionbaustein FB_VN_GevCameraControl [▶ 1672] können Sie die Kalibrierungsparameter in die SPS laden:

Variablen

```
hr                      :   HRESULT;
fbCamera                :   FB_VN_GevCameraControl;
aCameraMatrix           :   TcVnMatrix3x3_LREAL;
aDistortionCoefficients :   TcVnArray8_LREAL;
aRotationMatrix         :   TcVnMatrix3x3_LREAL;
aTranslationVector      :   TcVnVector3_LREAL;
```

Code

```
hr := fbCamera.GetCameraMatrix(aCameraMatrix);
hr := fbCamera.GetDistortionCoefficients(aDistortionCoefficients);
hr := fbCamera.GetRotationMatrix(aRotationMatrix);
hr := fbCamera.GetTranslationVector(aTranslationVector);
```

Schritt 7 – Bildpunkte in Weltkoordinaten transformieren

Anschließend bestimmen Sie zum Beispiel mit der Funktion F_VN_TransformCoordinatesImageToWorld_Points [▶ 1137] von einem Punkt im Bild die Koordinaten in der realen Welt in Bezug auf den Nullpunkt des Kalibriermusters:

Variablen

```
aPointImage               :   TcVnPoint2_LREAL;
aPointWorld               :   TcVnPoint3_LREAL
```

Code

```
hr := F_VN_TransformCoordinatesImageToWorld_Point(
        aSrcPoint                 :=  aPointImage,
        aDestPoint                :=  aPointWorld,
        aCameraMatrix             :=  aCameraMatrix,
        aDistortionCoefficients   :=  aDistortionCoefficients,
        aRotationMatrix           :=  aRotationMatrix,
        aTranslationVector        :=  aTranslationVector,
        fZ                        :=  0,
        hrPrev                    :=  hr);
```

Schritt 8 – Weltpunkte in Bildkoordinaten transformieren.

Ebenso kann diese Transformation mit der Funktion F_VN_TransformCoordinatesWorldToImage_Points [▶ 1141] umgekehrt erfolgen, sodass Punkte aus dem Weltkoordinatensystem so transformiert werden, dass sie an der korrekten Stelle im Bild dargestellt werden können. Dies ist sinnvoll, um Hilfslinien zur besseren Orientierung im Bild anzuzeigen. Im folgend dargestellten Code wird damit das Weltkoordinatensystem ins Bild eingezeichnet.

Variablen

```
i                         :   INT;
aLine                     :   TcVnVector4_DINT;
aCoordinatesWorld         :   ARRAY[0..4] OF TcVnPoint3_LREAL := [ [0,0,0], [50,0,0], [-50, 0, 0],
[0, 50, 0], [0, -50, 0] ];
aCoordinatesImage         :   ARRAY[0..4] OF TcVnPoint2_REAL;
```

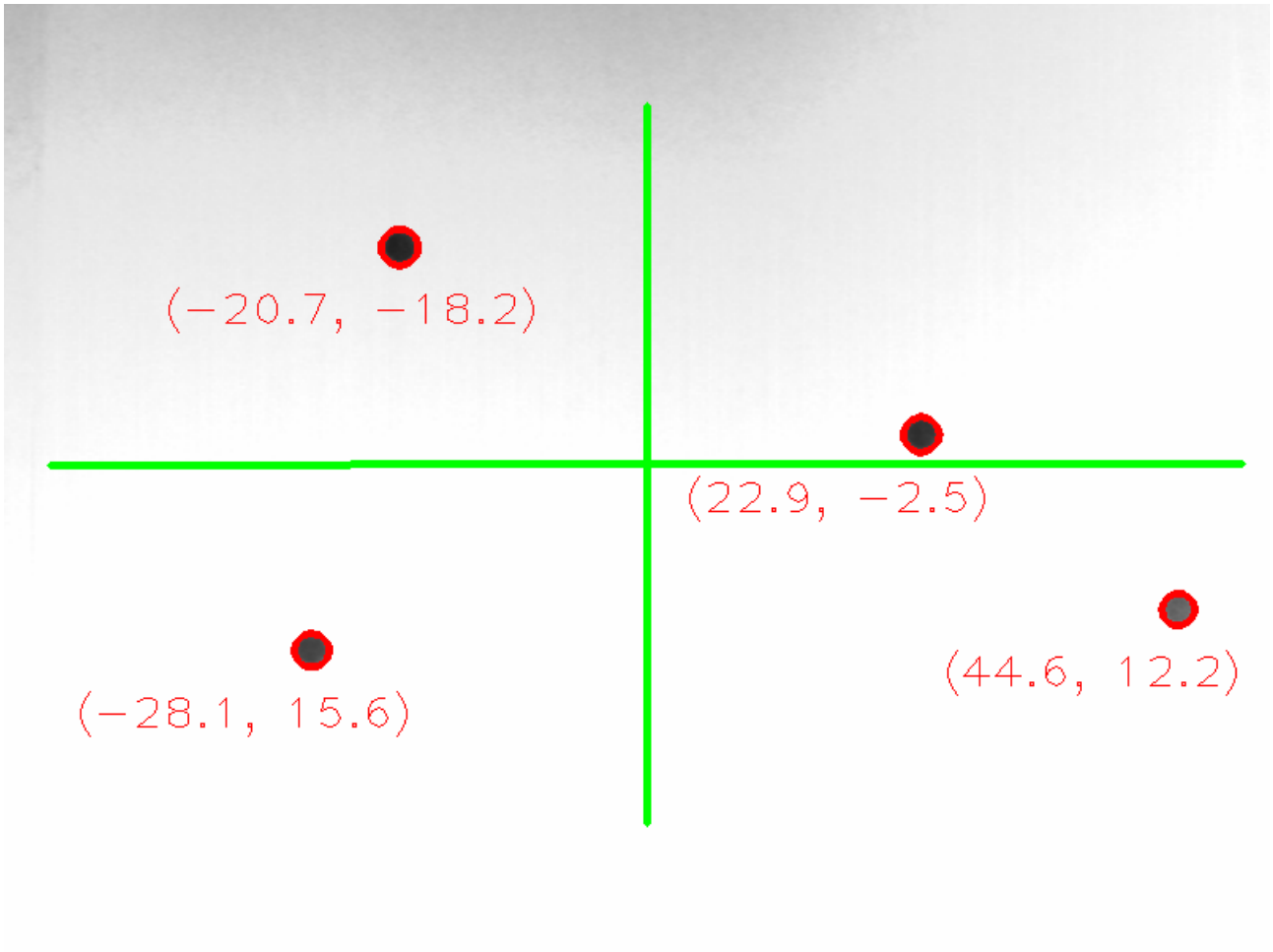
Code

```
FOR i:= 0 TO 4 DO
    hr := F_VN_TransformCoordinatesWorldToImage_Point(
            aSrcPoint                 :=  aCoordinatesWorld[i],
            aDestPoint                :=  aCoordinatesImage[i],
            aCameraMatrix             :=  aCameraMatrix,
            aDistortionCoefficients   :=  aDistortionCoefficients,
            aRotationMatrix           :=  aRotationMatrix,
            aTranslationVector        :=  aTranslationVector,
            hrPrev                    :=  hr);

    IF i > 0 THEN
        aLine[0] := REAL_TO_DINT(aCoordinatesImage[0][0]);
        aLine[1] := REAL_TO_DINT(aCoordinatesImage[0][1]);
        aLine[2] := REAL_TO_DINT(aCoordinatesImage[i][0]);
        aLine[3] := REAL_TO_DINT(aCoordinatesImage[i][1]);
    hr := F_VN_DrawLine_TcVnVector4_DINT(aLine , ipImageRes, aColorGreen, 3, hr);
    END_IF
END_FOR
```

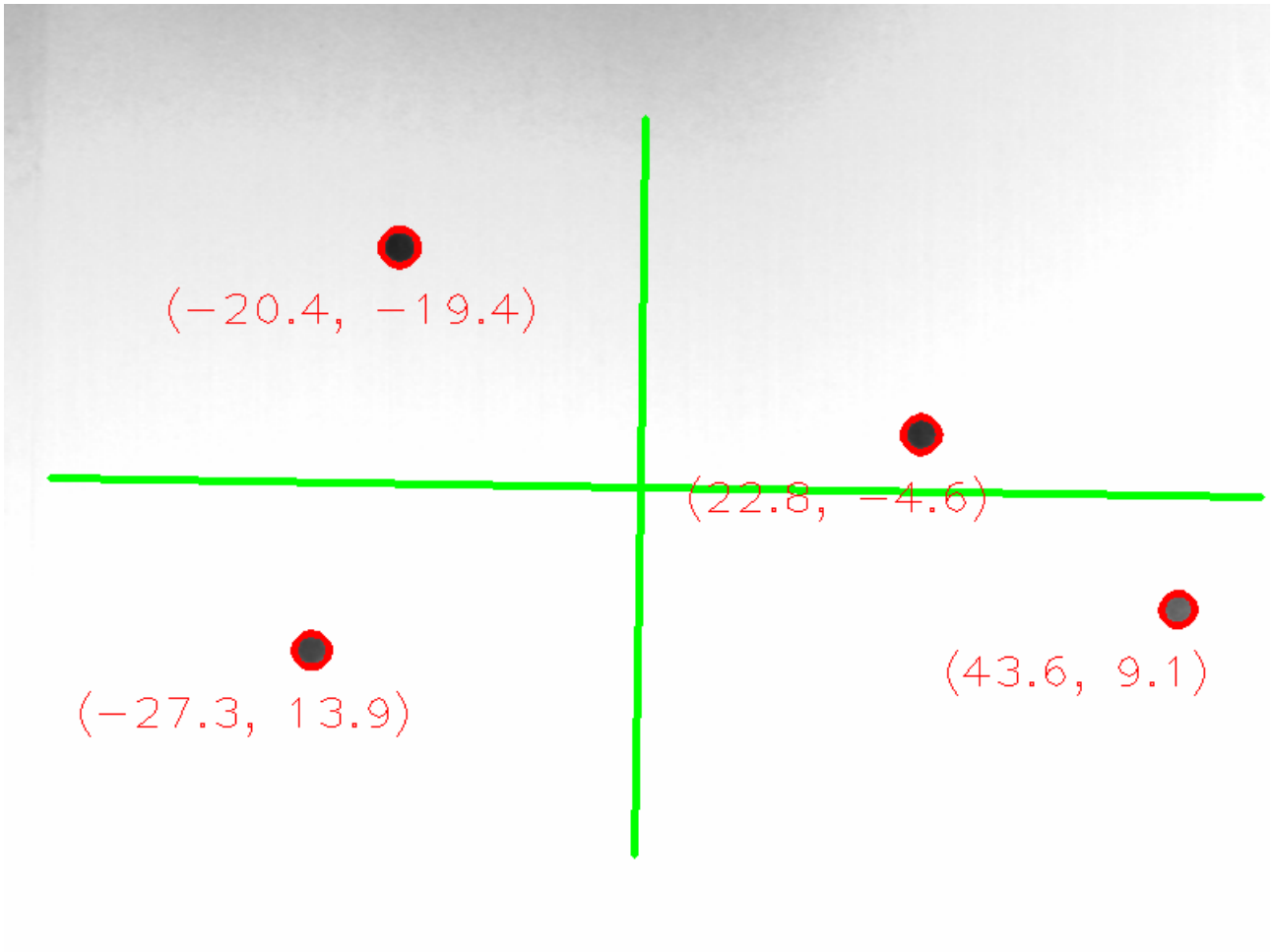
Schritt 9 – Ergebnisse ansehen

Wenn das Programm ausgeführt wird, muss zunächst noch das Abspielen des Kamera-Streams [▶ 112] gestartet werden. Öffnen Sie dazu den Karteireiter **Record/Playback** des Kamera-Objekts, wählen die CameraSimulation.tcs Datei im entsprechenden Bildordner aus und starten das Playback.

Danach kann das Ergebnisbild im ADS Image Watch [▶ 141] begutachtet werden. Man erkennt die eingezeichneten Koordinatenachsen und die ermittelten Weltkoordinaten der einzelnen Punkte auf dem Bild.



Wenn Sie bei der extrinsischen Kalibrierung in Schritt 4 ein anderes Bild verwenden oder einen anderen Koordinatenursprung verwenden, sieht das Ergebnis entsprechend anders aus. Der Koordinatenachsen sind verschoben und die Weltkoordinaten der Bildpunkte sind auch angepasst.



8.4 Beispiele zur Kamera-Konfiguration

Da die Parameter jeder GigE Vision Kamera sich unterscheiden können, ist keine vollständige Beschreibung möglich. Im Folgenden wird exemplarisch ein Überblick gegeben über die häufig vorkommenden Kamera-Parameter und wie diese zu konfigurieren sind. Eine Anleitung zur Konfiguration von GenAPI-Parametern finden Sie im Kapitel Konfiguration [▶ 83].

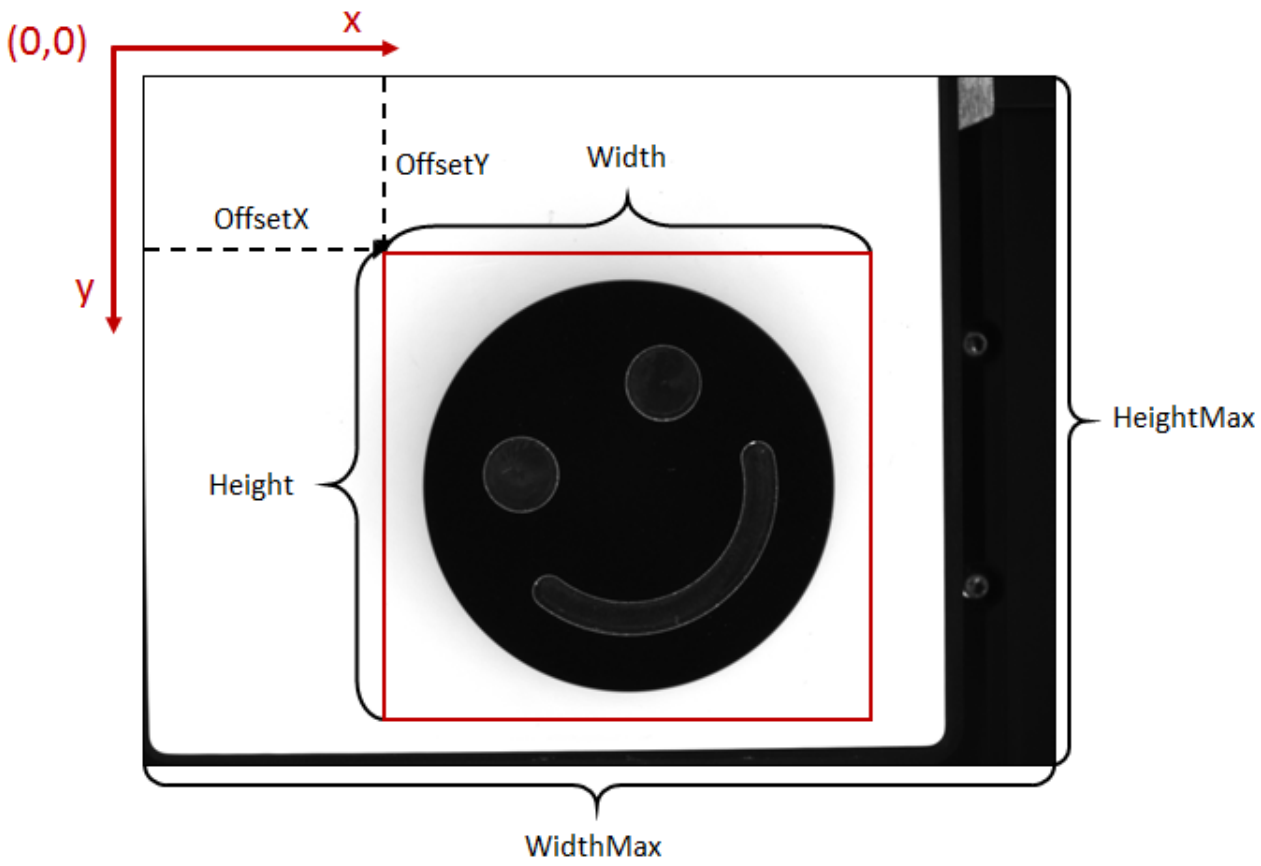
8.4.1 Region of Interest (ROI)

Nimmt der Kamerasensor ein größeres Sichtfeld auf als benötigt wird, empfiehlt es sich eine Region of Interest (ROI) auf das tatsächlich benötigte Sichtfeld zu legen:

- Abhängig vom Kamerasensor und ausgewählten Bereich, kann dadurch das Auslesen des Sensors schneller erfolgen.
- Die zu übertragende Datenmenge wird reduziert, wodurch sich auch die Datenübertragungszeit reduziert.
- Gleichzeitig reduziert sich auch die Verarbeitungszeit aller Bildverarbeitungsalgorithmen, welche über das gesamte Bild erfolgen.
- Zusätzlich können Störungen im Bild reduziert werden, wodurch die Bildverarbeitung robuster und einfacher werden kann.

Features der Kamera kontrollieren

Ob Ihre Kamera entsprechende Features unterstützt, entnehmen Sie dem Handbuch der Kamera. Gleiches gilt auch für die tatsächlich verwendeten Feature-Namen. Im Weiteren wird die Vorgehensweise mit den Feature-Namen entsprechend der GenICam Standard Features Naming Convention (Version 2.4) beschrieben.



Unter `ImageFormatControl` bzw. der entsprechenden Kategorie bei Ihrer Kamera befinden sich die Featrures zum Setzen einer ROI:

- `WidthMax` und `HeightMax` geben die maximale Breite und Höhe des Bildes in Pixeln an. Abhängig von weiteren Features (z. B. einem Binning [▶ 2933]) können diese Werte variieren.
- Im Gegenzug dazu geben `Width` und `Height` die Breite und Höhe der ROI, also des zu betrachtenden Sichtfeldes an und sind entsprechend zu setzen.
- Über `OffsetX` und `OffsetY` wird die ROI, bzw. das Sichtfeld positioniert. Startpunkt dabei ist die obere linke Bildecke.

Beim Setzen der Features ist zu beachten, dass:

- `OffsetX` + `Width` <= `WidthMax`
- `OffsetY` + `Height` <= `HeightMax`

8.4.2 Binning

Als Binning wird das Aufsummieren oder das Mittelwertbilden von mehreren Pixelwerten zu einem bezeichnet.

Es kann vorgenommen werden um

- die Lichtempfindlichkeit zu erhöhen,
- den Kontrast zu verbessern,
- die Bildauflösung bei gleichem Sichtfeld zu reduzieren, sodass schnellere Frameraten und Verarbeitungszeiten möglich sind.

Es sollte nicht vorgenommen werden, wenn durch die Reduzierung der Bildauflösung die geforderten Genauigkeiten nicht mehr erreicht werden können.

Beim Aufsummieren oder beim Mittelwertbilden der Pixel wird zwischen horizontalem und vertikalem Binning unterschieden.

Features der Kamera kontrollieren

Ob Ihre Kamera entsprechende Features unterstützt, entnehmen Sie dem Handbuch der Kamera. Gleiches gilt auch für die tatsächlich verwendeten Feature-Namen. Im Weiteren wird die Vorgehensweise mit den Feature-Namen entsprechend der GenICam Standard Features Naming Convention (Version 2.4) beschrieben.

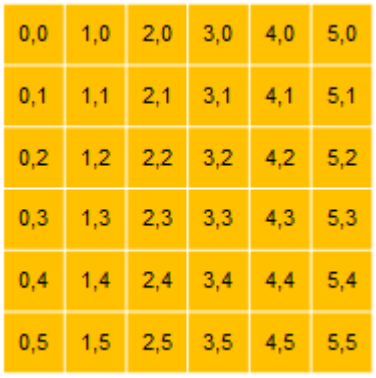
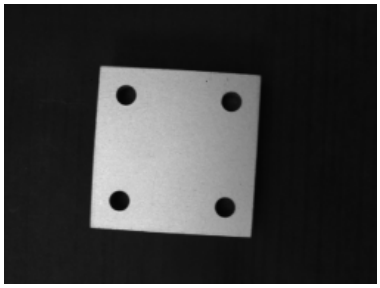
Unter `ImageFormatControl` bzw. der entsprechenden Kategorie bei Ihrer Kamera befinden sich die Features zum Setzen eines Binnings:

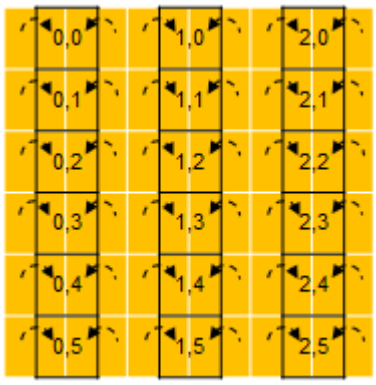
- `BinningSelector` gibt an, ob das Binning für den gesamten `Sensor` oder eine ROI [▶ 2932] gilt.
- `BinningHorizontalMode` und `BinningVerticalMode` geben an, ob die Pixel aufsummiert oder Mittelwerte gebildet werden sollen.

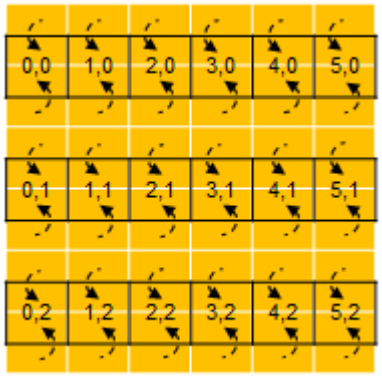
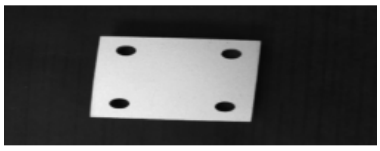
`BinningHorizontal` und `BinningVertical` geben an, wie viele Pixel jeweils in horizontaler Richtung und in vertikaler Richtung zusammengefasst werden sollen. Ein Wert von 1 gibt dabei an, dass kein Binning in die entsprechende Richtung erfolgt.

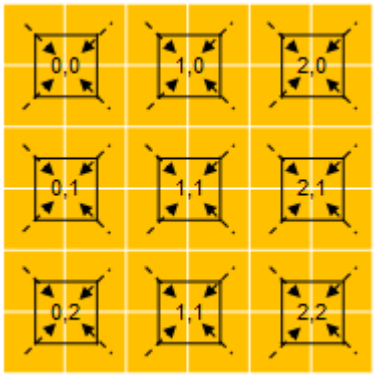
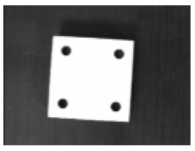
Binning bei einer monochromen Kamera

Hier exemplarisch vier mögliche Konfigurationen bei einer monochromen Kamera (dargestellt sind die Koordinatenwerte der Pixel):

`BinningHorizontal = 1`
`BinningVertical = 1`
⇒ Kein Zusammenfassen von Pixeln
		

`BinningHorizontal = 2`
`BinningVertical = 1`
`BinningHorizontalMode = Sum`
		

`BinningHorizontal = 1`
`BinningVertical = 2`
`BinningVerticalMode = Sum`
		

`BinningHorizontal = 2`
`BinningVertical = 2`
`BinningHorizontalMode = Sum`
`BinningVerticalMode = Sum`
		

Binning bei einer Farb-Kamera

Binning wird von Farb-Kameras seltener unterstützt als von monochromen Kameras und kann dort auf verschiedene Art und Weise umgesetzt sein. Die genaue Umsetzung ist dem Benutzerhandbuch der Kamera zu entnehmen.

8.4.3 Bild-Akquise und Trigger

Die Akquisition- und Trigger-Features sind miteinander verzahnt und werden daher im Weiteren zusammen betrachtet. Der Begriff `Acquisition` beschreibt grundsätzlich einen Zustand von Kameras,siehe ImageAcquisition [▶ 1662], in dem Bilder aufgenommen werden können. Ob und zu welchem Zeitpunkt tatsächlich Bilder aufgenommen und gesendet werden, wird durch die Konfiguration der `Trigger`-Parameter festgelegt.

Die verschiedenen Konfigurationsmöglichkeiten und Zusammenhänge zwischen Acquisition- und Trigger-Einstellungen werden in den nachfolgenden Kapiteln genauer beschrieben, die nach dem `AcquisitionMode` gegliedert sind.

Die mit Abstand am häufigsten genutzten Möglichkeiten der Bildaufnahme sind:

- Kontinuierliche Bild-Akquise [▶ 2938] – die Kamera streamt, sie nimmt ohne weitere Trigger in gleichmäßigen Abständen Bilder auf.
- Einzel-Bild-Trigger bei kontinuierlicher Bild-Akquise [▶ 2938] - die Kamera nimmt nach einem Trigger-Signal (Hardware oder Software) genau ein Bild auf. Die Kamera verbleibt dabei im Zustand `ACQUIRING`, daher ist es ein wiederholtes Schreiben des `AcquisitionStart` -Commands vor jeder Aufnahme ist nicht notwendig.

Features der Kamera kontrollieren

Ob Ihre Kamera entsprechende Features unterstützt, entnehmen Sie dem Handbuch der Kamera. Gleiches gilt auch für die tatsächlich verwendeten Feature-Namen. Im Weiteren wird die Vorgehensweise mit den Feature-Namen entsprechend der GenICam Standard Features Naming Convention (Version 2.4) beschrieben.

Acquisition Mode

Das `AcquisitionMode` Feature gibt hauptsächlich an, wie viele Bilder während einer Akquisition aufgenommen werden und bezieht sich auf den `TCVN_CS_ACQUIRING`-State der Kamera Statemachine. Zum Beenden einer Aufnahme muss der State durch ein `AcquisitionStop`-Command verlassen werden, bevor die nächste Aufnahme wieder mit einem `AcquisitionStart`-Command beginnen kann. Daher ist der `Continuous`-Mode die am häufigsten gewählte Einstellung, da man dann dauerhaft in dem Acquiring-State bleiben kann.

- `Continuous`
  - Die Bildaufnahme beginnt mit dem `AcquisitionStart`-Command und wird erst durch das `AcquisitionStop`-Command beendet. Dazwischen werden je nach Trigger-Einstellungen kontinuierlich Bilder aufgenommen.
  - Kontinuierliche Bild-Akquise [▶ 2938]
- `MultiFrame`
  - Bis die in `AcquisitionFrameCount` vorgegebene Bildanzahl erreicht ist, werden Bilder aufgenommen. Für jede Bildaufnahmesequenz ist ein `AcquisitionStart`-Command erforderlich.
  - Multi-Bild-Akquise [▶ 2939]
- `SingleFrame`
  - Ein Bild wird aufgenommen. Für jede Bildaufnahme ist ein `AcquisitionStart`-Command erforderlich.
  - Einzelbild-Akquise [▶ 2941]

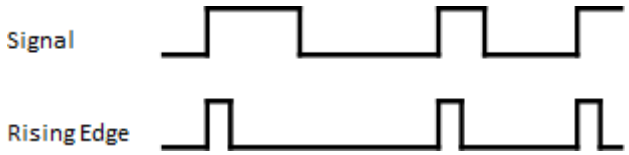
Trigger Source

Das Feature `TriggerSource` gibt an, welches interne oder physikalische Signal als Trigger interpretiert werden soll. Die Liste der möglichen Quellen ist lang und variiert je nach Kamera, häufige sind:

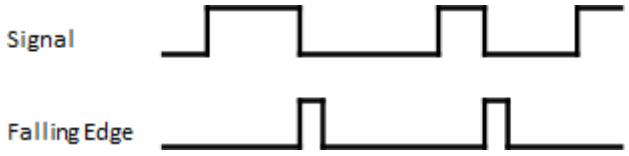
- `Software`
  Ein Software-Kommando stellt den Trigger da. Die SPS-Funktionsbausteine FB_VN_GevCameraControl [▶ 1672] und FB_VN_SimpleCameraControl [▶ 1697] stellen zum Senden des Kommandos die Methode TriggerImage [▶ 1696] bereit.
- `Line0, Line1, ...`
  Ein physikalischer Kameraeingang ist die Triggerquelle. Bei der daran angeschlossenen Signalquelle achten Sie auf die notwendige Flankensteilheit. Diese kann je nach Kamera variieren.
- `Encoder0, Encoder1, ...`
  Zeilenkameras werden oft mit einem Encoder verbunden, um die Bildaufnahme zu triggern. Steht kein physikalischer Encoder zur Verfügung, aber die entsprechenden Achsenwerte liegen in TwinCAT vor, kann das Encoder Signal über eine EL2521 bzw. EL2522 ausgegeben werden.

Trigger Activation

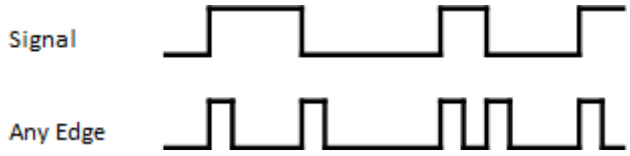
- `RisingEdge`
  Der Trigger wird bei einer steigenden Flanke der `TriggerSource` beachtet.



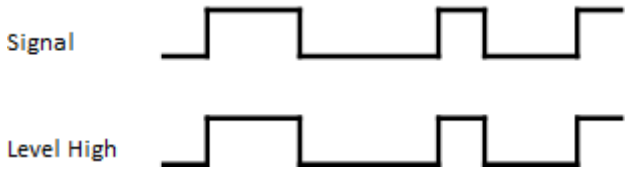
- `FallingEdge`
  Der Trigger wird bei einer fallenden Flanke der `TriggerSource` beachtet.



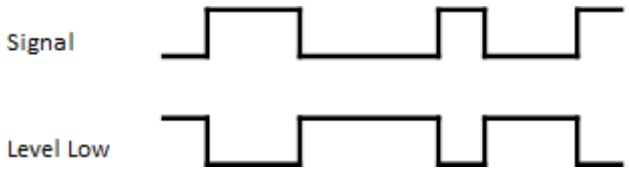
- `AnyEdge`
  Der Trigger wird sowohl bei einer steigenden als auch bei einer fallenden Flanke der `TriggerSource` beachtet.



- `LevelHigh`
  Der Trigger wird beachtet, solang ein Signal an der `TriggerSource` anliegt.



- `LevelLow`
  Der Trigger wird beachtet, solang kein Signal an der `TriggerSource` anliegt.



8.4.3.1 Kontinuierliche Bild-Akquise

Die Einstellungen zur Bild-Akquise und zum Triggern sind standardmäßig miteinander verzahnt. Im folgenden werden einige Zusammenhänge erläutert, wobei davon ausgegangen wird, dass der `AcquisitionMode` auf `Continuous` gesetzt ist.

Die anderen Fälle sind beschrieben in:

- Multi-Bild-Akquise [▶ 2939]
- Einzelbild-Akquise [▶ 2941]

Features der Kamera kontrollieren

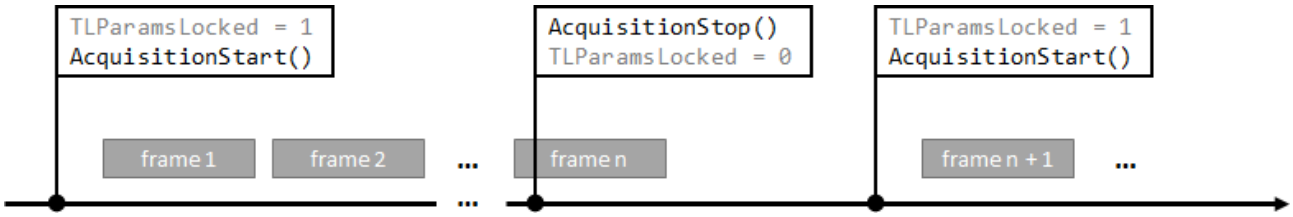
Ob Ihre Kamera entsprechende Features unterstützt, entnehmen Sie dem Handbuch der Kamera. Gleiches gilt auch für die tatsächlich verwendeten Feature-Namen. Im Weiteren wird die Vorgehensweise mit den Feature-Namen entsprechend der GenICam Standard Features Naming Convention (Version 2.4) beschrieben.

Kontinuierliche Bild-Akquise

Soll die Kamera streamen, das heißt ohne weitere Trigger in gleichmäßigen Abständen Bilder aufnehmen, ist folgende Konfiguration erforderlich:

- `AcquisitionMode = Continuous`
- `TriggerSelector = AcquisitionStart`
- `TriggerMode = Off`

Durch das `AcquisitionStart`-Command wird dann das Streamen gestartet und durch das `AcquisitionStop`-Command beendet, wobei die Aufnahme des aktuellen Bildes noch beendet und übertragen wird.

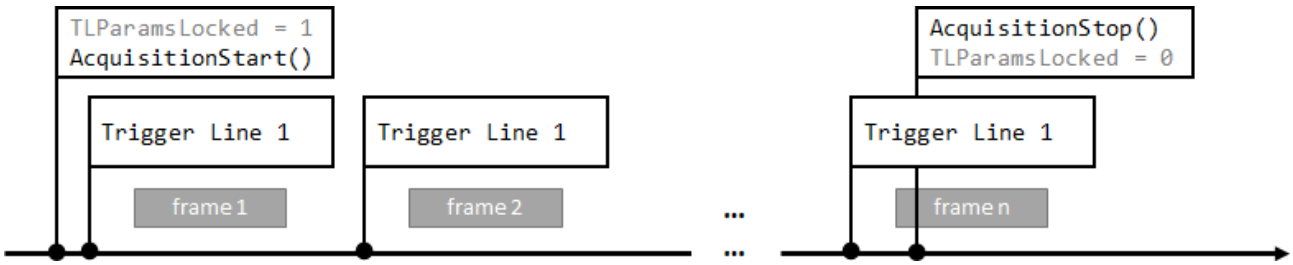


Einzel-Bild-Trigger bei kontinuierlicher Bild-Akquise

Soll die Kamera nach einem Trigger-Signal genau ein Bild aufnehmen, ist folgende Konfiguration erforderlich:

- `AcquisitionMode = Continuous`
- `TriggerSelector = FrameStart`
- `TriggerMode = On`
- `TriggerSource = Line1 (exemplarisch)`

Wie in den anderen Fällen ist die Kamera nach dem `AcquisitionStart`-Command bereit Bilder aufzunehmen. Die Aufnahme jedes einzelnen Bildes erfolgt jeweils nach einem Trigger-Signal der ausgewählten Quelle. Wie das Trigger-Signal auszusehen hat, wird in `TriggerActivation` angegeben.

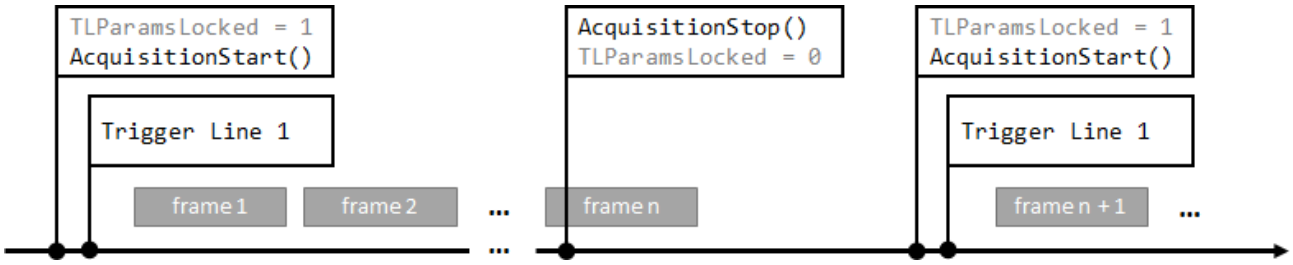


Kontinuierliche Bild-Akquise nach Trigger-Start-Signal

Soll die Kamera nach einem Trigger Signal streamen, das heißt in gleichmäßigen Abständen Bilder aufnehmen, ist folgende Konfiguration erforderlich:

- `AcquisitionMode = Continuous`
- `TriggerSelector = AcquisitionStart`
- `TriggerMode = On`
- `TriggerSource = Line1 (exemplarisch)`

Nach dem `AcquisitionStart`-Command ist die Kamera bereit Bilder aufzunehmen und zu übertragen. Die Aufnahme beginnt aber erst nach einem Trigger-Signal, im Beispiel nach einem Hardware-Trigger-Signal auf Line 1. Durch das `AcquisitionStop`-Command wird das Streamen beendet, wobei die Aufnahme des aktuellen Bildes noch beendet und übertragen wird.

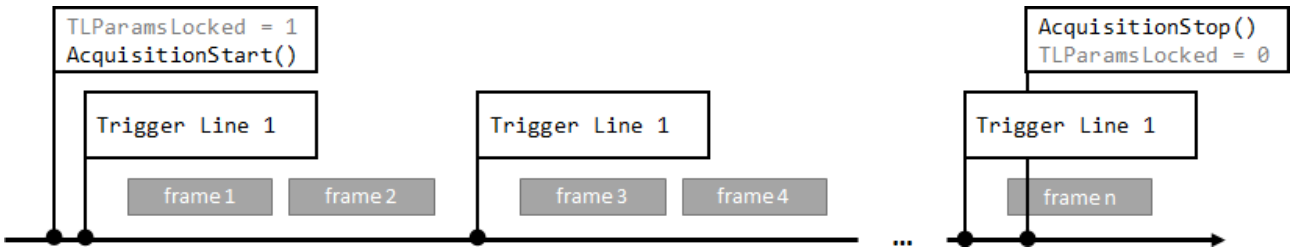


Multi-Bild-Trigger bei kontinuierlicher Bild-Akquise

Soll die Kamera nach einem Trigger Signal eine bestimmte Anzahl Bilder aufnehmen, ist folgende Konfiguration erforderlich, wobei exemplarisch die Anzahl auf 2 gesetzt ist:

- `AcquisitionMode = Continuous`
- `AcquisitionBurstFrameCount = 2 (exemplarisch)`
- `TriggerSelector = FrameBurstStart`
- `TriggerMode = On`
- `TriggerSource = Line1 (exemplarisch)`

Wie in den anderen Fällen ist die Kamera nach dem `AcquisitionStart`-Command bereit Bilder aufzunehmen. Die Aufnahme der definierten Anzahl an Bildern erfolgt jeweils nach einem Trigger-Signal der ausgewählten Quelle. Wie das Trigger-Signal auszusehen hat, wird in `TriggerActivation` angegeben. Erfolgt das `AcquisitionStop`-Command während der Bildaufnahme/-übertragung wird das bereits begonnene Bild noch beendet, evtl. noch fehlende Bilder aber nicht mehr erstellt.



8.4.3.2 Multi-Bild-Akquise

Die Einstellungen zur Bild-Akquise und zum Triggern sind standardmäßig miteinander verzahnt. Im folgenden werden einige Zusammenhänge erläutert, wobei davon ausgegangen wird, dass der `AcquisitionMode` auf `MultiFrame` gesetzt ist.

Die anderen Fälle sind beschrieben in:

- Kontinuierliche Bild-Akquise [▶ 2938]
- Einzelbild-Akquise [▶ 2941]

Features der Kamera kontrollieren

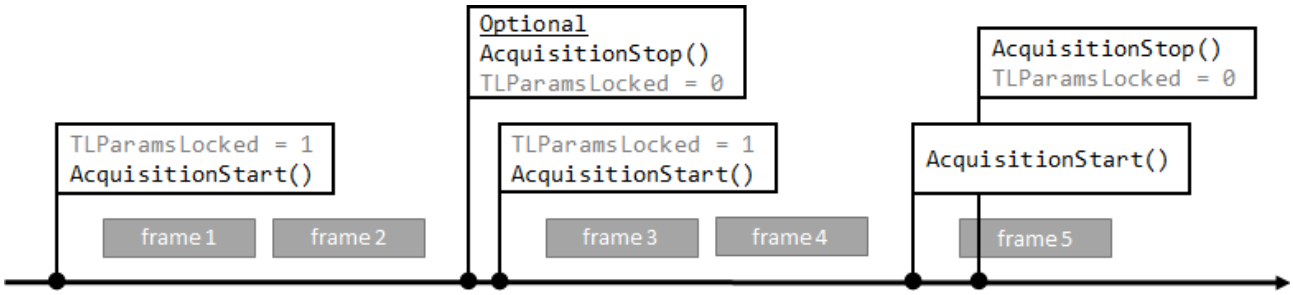
Ob Ihre Kamera entsprechende Features unterstützt, entnehmen Sie dem Handbuch der Kamera. Gleiches gilt auch für die tatsächlich verwendeten Feature-Namen. Im Weiteren wird die Vorgehensweise mit den Feature-Namen entsprechend der GenICam Standard Features Naming Convention (Version 2.4) beschrieben.

Multi-Bild-Akquise

Soll die Kamera eine bestimmte Anzahl an Bildern streamen, ohne die Verwendung eines Triggers, ist folgende Konfiguration erforderlich:

- `AcquisitionMode = MultiFrame`
- `AcquisitionFrameCount = 2 (exemplarisch)`
- `TriggerSelector = AcquisitionStart`
- `TriggerMode = Off`

Durch das `AcquisitionStart`-Command wird das Streamen gestartet und beim Erreichen der vorgegebenen Bildanzahl (`AcquisitionFrameCount`) beendet. Optional kann danach das `AcquisitionStop`-Command und die Aufhebung des Locks erfolgen. Dies ermöglicht es, vor dem nächsten `AcquisitionStart`-Command Kamerafeatures zu ändern. Erfolgt das `AcquisitionStop`-Command während der Bildaufnahme/-übertragung, wird das bereits begonnene Bild noch beendet, evtl. noch fehlende Bilder aber nicht mehr erstellt.

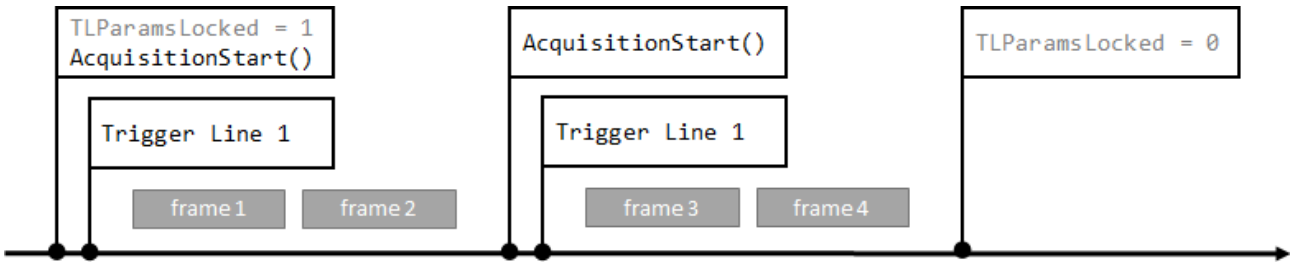


Multi-Bild-Akquise nach Trigger-Start-Signal

Soll die Kamera nach einem Trigger-Signale eine bestimmte Anzahl an Bildern streamen, ist folgende Konfiguration erfoderlich:

- `AcquisitionMode = MultiFrame`
- `AcquisitionFrameCount = 2 (exemplarisch)`
- `TriggerSelector = AcquisitionStart`
- `TriggerMode = On`
- `TriggerSource = Line1 (exemplarisch)`

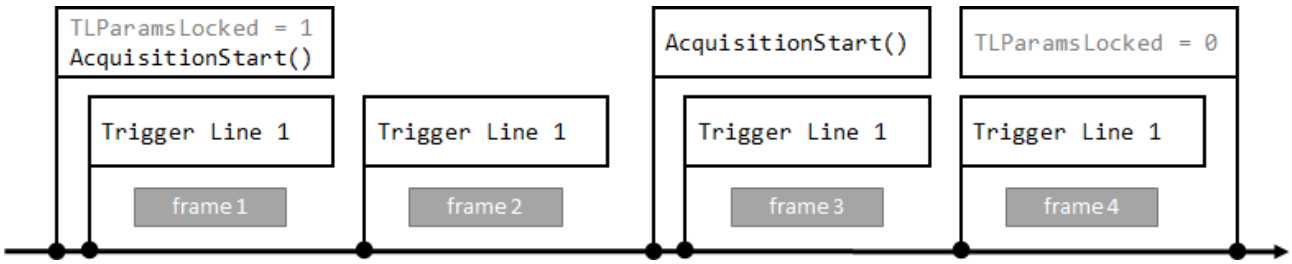
Durch das `AcquisitionStart`-Command ist die Kamera bereit zu streamen. Die Aufnahme der vorgegebenen Bilder beginnt aber erst nach einem Trigger-Signal, im Beispiel nach einem Hardware Trigger-Signal auf Line 1. Nach dem Erreichen der vorgegebenen Bildanzahl kann wie bei der Multi-Bild-Akquise (ohne Trigger) optional das `AcquisitionStop`-Command und die Aufhebung des Locks erfolgen.



Multi-Bild-Akquisition mit Einzel-Bild-Trigger

Konfiguration:

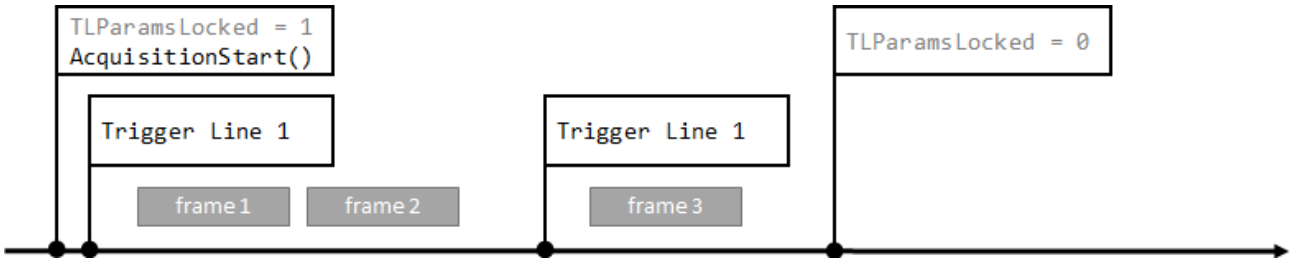
- `AcquisitionMode = MultiFrame`
- `AcquisitionFrameCount = 2 (exemplarisch)`
- `TriggerSelector = FrameStart`
- `TriggerMode = On`
- `TriggerSource = Line1 (exemplarisch)`



Multi-Bild-Akquisition mit Multi-Bild-Trigger

Konfiguration:

- `AcquisitionMode = MultiFrame`
- `AcquisitionFrameCount = 3 (exemplarisch)`
- `AcquisitionFrameBurstCount = 2 (exemplarisch)`
- `TriggerSelector = FrameBurstStart`
- `TriggerMode = On`
- `TriggerSource = Line1 (exemplarisch)`



8.4.3.3 Einzelbild-Akquise

Die Einstellungen zur Bild-Akquise und zum Triggern sind standardmäßig miteinander verzahnt. Im folgenden werden einige Zusammenhänge erläutert, wobei davon ausgegangen wird, dass der `AcquisitionMode` auf `SingleFrame` gesetzt ist.

Die anderen Fälle sind beschrieben in:

- Kontinuierliche Bild-Akquise [▶ 2938]
- Multi-Bild-Akquise [▶ 2939]

Features der Kamera kontrollieren

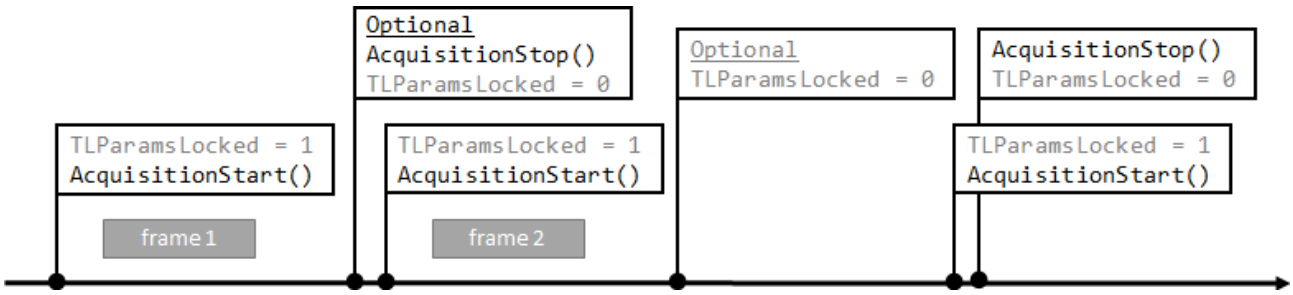
Ob Ihre Kamera entsprechende Features unterstützt, entnehmen Sie dem Handbuch der Kamera. Gleiches gilt auch für die tatsächlich verwendeten Feature-Namen. Im Weiteren wird die Vorgehensweise mit den Feature-Namen entsprechend der GenICam Standard Features Naming Convention (Version 2.4) beschrieben.

Einzelbild-Akquise

Soll nach jedem `AcquisitionStart`-Command genau ein Bild aufgenommen werden, ist folgende Konfiguration erforderlich:

- AcquisitionMode = SingleFrame
- TriggerSelector = AcquisitionStart
- TriggerMode = Off

Nach dem AcquisitionStart-Command wird genau ein Bild aufgenommen. Zwischen zwei AcquisitionStart-Commands kann ein AcquisitionStop-Command, sowie die Aufhebung des Locks erfolgen. Die Aufhebung des Locks ermöglicht es, Kamerafeatures zwischen den AcquisitionStart-Commands zu ändern. Erfolgt ein AcquisitionStop-Command so kurz nach dem AcquisitionStart-Command, dass noch nicht mit der Bildaufnahme begonnen wurde, so wird kein Bild aufgenommen.



Einzelbild-Akquise mit Bildaufnahme-Trigger
oder Einzelbild-Akquise mit Einzelbild-Trigger

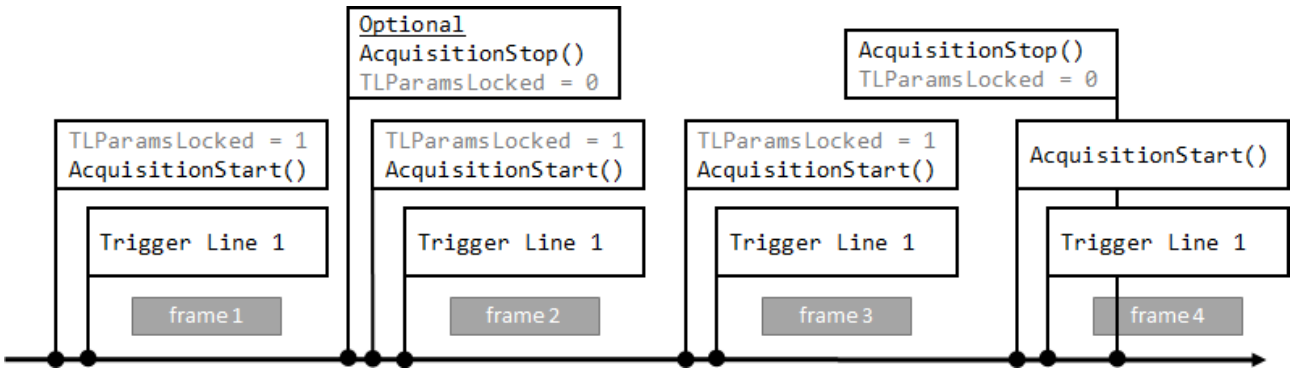
Soll nach jedem AcquisitionStart-Command genau ein Bild aufgenommen werden, die Aufnahme aber erst nach einem Trigger-Signal erfolgen, sind folgende Konfigurationen möglich:

Konfiguration - Einzelbild-Akquise mit Bildaufnahme-Trigger

- AcquisitionMode = SingleFrame
- TriggerSelector = AcquisitionStart
- TriggerMode = On
- TriggerSource = Line1 (exemplarisch)

Konfiguration - Einzelbild-Akquise mit Einzel-Bild-Trigger

- AcquisitionMode = SingleFrame
- TriggerSelector = FrameStart
- TriggerMode = On
- TriggerSource = Line1 (exemplarisch)



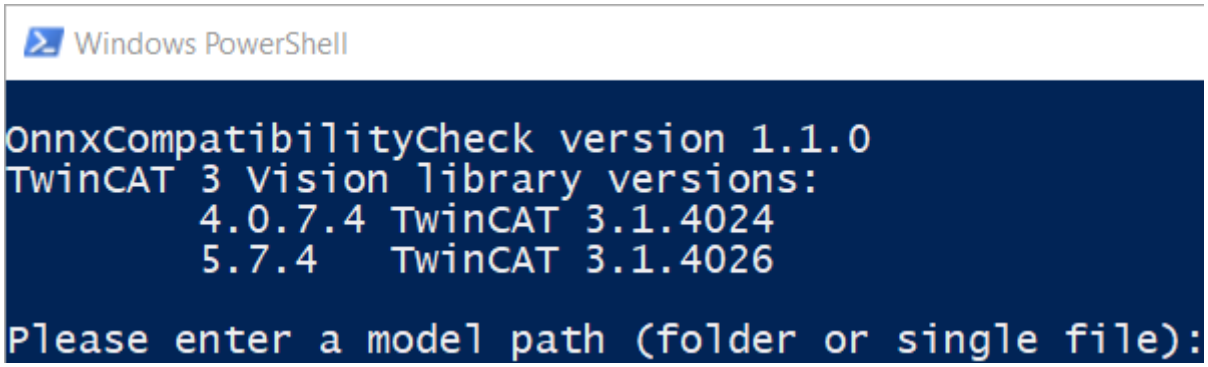
8.5 Neural Network Compatibility Check

Dieses Tool bietet die Möglichkeit die Kompatibilität von ONNX-Modellen zu den Funktionen des Produkts „TF7810 | TwinCAT 3 Vision Neural Network“ zu prüfen. Das „TF7810_NeuralNetworkSupplements.zip“ kann direkt von der Beckhoff Webseite heruntergeladen werden.

Das Paket beinhaltet die ausführbare Datei „OnnxCompatibilityCheck.exe“, zur vereinfachten Anwendung das Power Shell Skript „CheckModelCompatibility.ps1“ und einen Ordner „ONNX_Samples“ mit einem Beispielmodell.

Ausführung des Power Shell Skripts

Nach dem Ausführen des Skripts werden zunächst die Versionsinformationen ausgegeben. Zuerst von der Anwendung selbst und dann zu den TwinCAT 3 Vision Bibliotheksversionen, auf die sich die Kompatibilität bezieht. Als Anwender werden Sie aufgefordert, einen Ordner oder Modellpfad anzugeben. Wird ein Ordner angegeben, in dem sich mehrere ONNX-Modelle befinden, werden diese in alphabetischer Reihenfolge getestet. Es ist auch möglich, explizit ein Modell anzugeben, wie z.B. "C:\ONNX_Samples\LemonModel.onnx“.



Eine erfolgreiche Ausführung sieht, z.B. anhand des Lemon Sample Models, wie folgt aus:

```
Please enter a model path (folder or single file): C:\ONNX_Samples\LemonModel.onnx

Successfully loaded 'LemonModel.onnx'
Model inputs: 1
  'input.1' [1, 3, 224, 224], TCVN_DL_4D_NCHW
Model Outputs: 1
  '367' [1, 3]
The model can be executed with TwinCAT Vision.
```

Am Ende wird die Ausführung pausiert, um sich die Ausgaben anschauen zu können.

Optionen des Power Shell Skripts

Wenn der Anwender das Skript zum Bearbeiten öffnet, findet er die Parameter $modelPath und $inputShape in Zeile 2 und 3. Der erste Parameter $modelPath ist per Default leer, so dass der Anwender zur Eingabe des Pfads aufgefordert wird. Alternativ kann im Skript ein Ordner- oder ein Modell-Pfad fest hinterlegt werden. Dann erhält der Anwender keine Aufforderung zur Eingabe und es wird direkt der hinterlegte Pfad verwendet.

CheckModelCompatibility.ps1

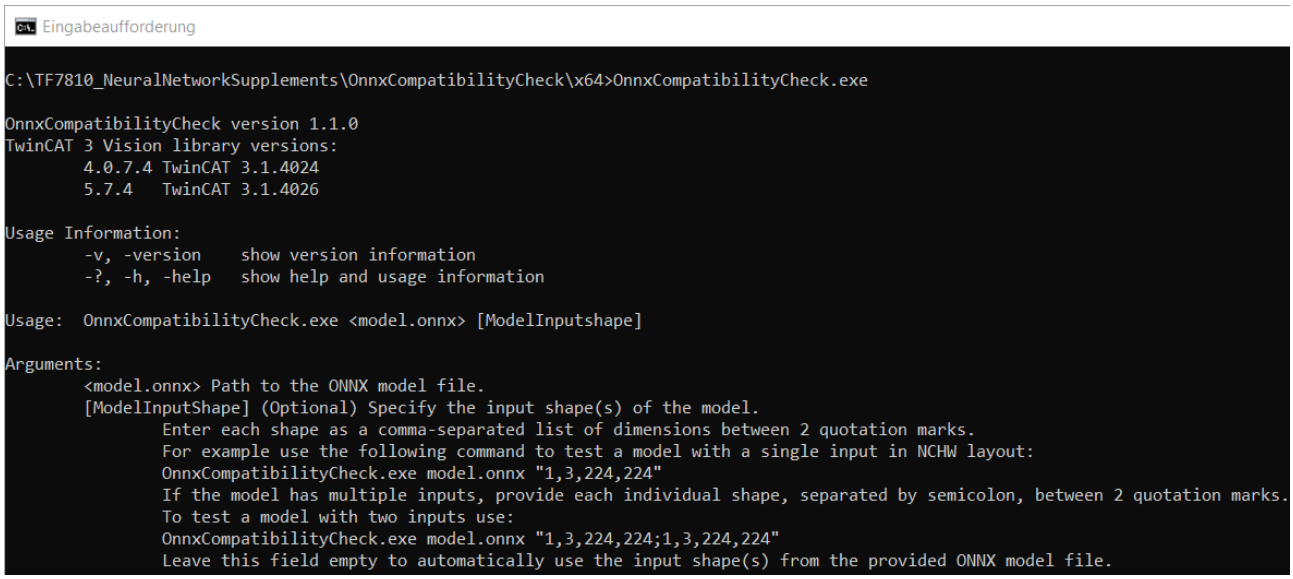
```
1  param(
2      [string]$modelPath = "", # folder or single file, e.g. "C:\temp\model.onnx"
3      [string]$inputShape = "skip" # "skip" if not required or model input shape, e.g. "1,3,224,224;1,3,32,32" for 2 inputs
4  )
```

Per Default ist für den zweiten optionalen $inputShape Parameter das Wort „skip“ hinterlegt, so dass diese Abfrage nicht erscheint und die Anwendung ohne die Option ausgeführt wird. Hier kann ebenfalls ein fester Wert, wie z.B. „1 3 224 224“ für einen einzelnen Input angegeben werden. Bei mehreren Inputs müssen die Angaben durch Semikolons getrennt werden z.B. „1,3,224,224;1,3,32,32“ für zwei Inputs. Alternativ kann das Wort „skip“ gelöscht werden, so dass der Anwender zur Eingabe aufgefordert wird. Diese Eingabeaufforderung zum „Model Input Shape“ kann bei der Ausführung des Skripts mit „Enter“ ebenfalls übersprungen werden.

Anwendung der „OnnxCompatibilityCheck.exe“

Die „OnnxCompatibilityCheck.exe“ kann direkt ohne Power Shell Skript ausgeführt werden. In der Windows Eingabeaufforderung sieht der erfolgreiche Test des Lemon Sample Modells z.B. wie folgt aus:

Wird nur der Name der Anwendung angegeben oder zusätzlich -?, -h, -help hinzugefügt und dann mit Enter bestätigt, werden die folgenden Informationen und Hinweise zur Anwendung ausgegeben.



Ergebnisausgaben

Im Folgenden sind die Ergebnisausgaben beschrieben, die bei unterschiedlichen Szenarien ausgegeben werden können:

- Zu Beginn der Ausführung wird geprüft, ob der Pfad gültig ist und die Datei im ONNX-Format vorliegt. Ist dies nicht der Fall, bricht die Ausführung ab und es wird eine entsprechende Meldung ausgegeben.
- Wenn das ONNX-Modell erfolgreich geladen werden konnte, wird dies als erste Information inklusive des Dateinamens ausgegeben.
- Danach folgen Informationen zu den Eingabe- und Ausgabeschichten des Modells, die aus der ONNX-Datei ausgelesen werden:
  - Für die Eingabeschichten wird zunächst die Anzahl ausgegeben, gefolgt von einer Aufzählung der Schichten mit der jeweiligen Bezeichnung und der Form des Tensors in eckigen Klammern. Angezeigt wird die Anzahl der Elemente in der jeweiligen Dimension.
  - Wenn eine Eingabeschicht mehr als zwei Dimensionen hat, wird zusätzlich noch der ETcVnDataLayout [▶ 236] für die Konvertierung (F_VN_ConvertDataLayout [▶ 1591]), von einem 2D-Bild zu dem jeweiligen Eingangsformat des Modells, ausgegeben. Als Vorrausetzungen für die korrekte Erkennung des `ETcVnElementType` wird angenommen, dass N (Sampleanzahl bzw. Batchgröße) = 1 ist und die Anzahl der Kanäle kleiner als die Breite oder Höhe ist.
  - Für die Eingangsschicht des Lemon Sample Models sieht das z.B. so aus: „Model inputs: 1 'input.1' [1, 3, 224, 224], TCVN_DL_4D_NCHW“.
  - Für ein Modell mit mehreren Eingabeschichten sieht das z.B. so aus: „Model inputs: 3 'input.0' [1, 3, 224, 224], TCVN_DL_4D_NCHW 'input1' [1, 5] 'input2' [1, 10]“
  - Für die Ausgabeschichten wird zunächst die Anzahl ausgegeben, gefolgt von einer Aufzählung der Schichten mit der jeweiligen Bezeichnung und der Form des Tensors. Angezeigt wird die Anzahl der Elemente in der jeweiligen Dimension.
  - Für die Ausgabeschicht des Lemon Sample Models sieht das z.B. so aus:
    „Model outputs: 1
    Layer: '367' [1, 3]“

- Neuronale Netze können entweder für ein festes (static) Eingangsformat ausgelegt sein oder die Möglichkeit bieten, mehrere unterschiedliche Eingangsformate (dynamic) verarbeiten zu können. Bei den meisten Modellen ist die Eingabeschicht fest definiert und die Werte des Tensors sind in der ONNX-Datei hinterlegt.
  Dann können die Werte ausgelesen, angezeigt und direkt zum Test verwendet werden.
- Wenn Modelle mehrere Eingangs- / Bildgrößen unterstützen, fehlen in diesen Fall die Angaben zu den Dimensionen wie z.B. Breite und Höhe des Bildes in der ONNX-Datei. Dann spricht man von einer dynamischen Form der Eingabeschicht. Wenn diese Form bei einer Eingabeschicht erkannt wird, dann wird hinter der Bezeichnung der Schicht, statt der Form des Tensors, das Wort „dynamic“ ausgegeben:
  - „'input.1' dynamic To test the execution, please provide the optional [ModelInputshape] (For 1 input use e.g. OnnxCompatibilityCheck.exe DynamicShape.onnx "1,3,224,224").”
- Um ein Modell mit dynamischer Eingabeschicht testen zu können, können ergänzend beim Aufruf der Anwendung die Werte für den „Model Input Shape“ übergeben werden. Ein Beispiel für den Aufruf und die Übergabe der Werte wird mit ausgegeben oder ist in der Hilfe zu sehen.
- Es gibt Modelle, bei denen die Eingabeschicht in der ONNX-Datei fest definiert ist, die aber auch mit anderen Werten ausgeführt werden können. Wenn der Anwender diese Information hat bzw. die weiteren Werte kennt, kann er ebenfalls den Aufruf mit der zusätzlichen Angabe des „Model Input Shape“ verwenden.
- Die Informationen zu den unterstützten Eingangsformaten können in der Regel in der Dokumentation des Modellanbieters, der Quelle des Modells oder im zugehörigen Forschungsartikel gefunden werden. Falls diese Informationen nicht explizit angegeben sind, können diese auch durch die Untersuchung der Modellarchitektur oder durch die Verwendung von Tools zur Inspektion der ONNX-Datei abgeleitet werden.
- Für den Fall, dass ein Modell nicht ausgeführt werden kann, wird neben der grundsätzlichen Information versucht, eine mögliche Ursache auszugeben. Falls die Ursache nicht eindeutig ist, wird keine zusätzliche Information ausgegeben. Mögliche Ursachen sind z.B. nicht bekannte oder unterstützte Operatoren / Datentypen oder eine neuere ONNX „opset number“. Wenn ein Operator als Ursache genannt wird, kann das auch ein Folgeproblem sein. Daher sollten auch die Operatoren der vorherigen Schicht analysiert werden.

9 Appendix

In diesem Kapitel finden Sie zusätzliche Informationen wie häufig gestellte Fragen, eine Übersicht der von TwinCAT Vision verwendeten Pfade und ein Kapitel zum Troubleshooting zur Abhilfe von bekannten Problemen.

9.1 FAQ

Was ist TwinCAT Vision?

TwinCAT Vision ist eine industrielle Bildverarbeitungssoftware. Sie kann Bilder mit Industriekameras aufnehmen und daraus Informationen über die aufgenommene Szene errechnen.

Häufige Anwendungsfälle sind:

- Anwesenheitskontrolle
- Objekterkennung
- Optische Vermessung
- Code Reading

Kann ich TwinCAT Vision kostenlos testen?

Ja, Sie können 7-Tage Testlizenzen für die Laufzeit-Komponenten erzeugen. Das Engineering ist kostenlos.

Benötige ich TwinCAT, um TwinCAT Vision zu nutzen?

Ja, TwinCAT Vision ist vollständig in TwinCAT integriert. Eine kompatible TwinCAT-Version muss installiert sein, bevor TwinCAT Vision installiert werden kann.

Welche Kameras kann ich mit TwinCAT Vision verwenden?

TwinCAT Vision unterstützt Kameras mit der GigE Vision Schnittstelle. GigE Vision Kameras werden über ein Ethernet-Kabel an den IPC angeschlossen.

Kann ich mehrere Kameras gleichzeitig verwenden?

Ja, mit dem TwinCAT GigE Vision Connector können Sie lizenz-abhängig mehrere Kameras betreiben.

Kann ich mehrere Kameras über einen einzigen Switch anschließen?

Ja. Berücksichtigen Sie dabei die maximal verfügbare Bandbreite (z. B. 125MB/s bei 1 GigE, effektiv ca. 110-115 MB/s).

Gibt es Beispiele für TwinCAT Vision?

Ja, im Kapitel Beispiele [▶ 2838].

Gibt es Benchmarks für die TwinCAT Vision Funktionen?

Nein, denn die Ausführungszeit von Vision-Algorithmen hängt stark von der Leistungsfähigkeit des IPCs, der Bildgröße, den Funktionsparametern und dem Bildinhalt ab. Daher ist es hilfreich, ein grundlegendes Verständnis dafür zu haben, wie Algorithmen funktionieren und welche Einflussfaktoren die jeweilige Rechenzeit beeinflussen.

Kann alternativ zu TwinCAT Vision auch andere Software zur Bildverarbeitung in TwinCAT genutzt werden?

Mit dem TC3 GigE Vision Connector können Bilder über GigE Vision aufgenommen werden. Die Bildverarbeitung kann dann auf verschiedene Weise erfolgen:

- Nutzung der TwinCAT Vision API über die SPS-Bibliothek Tc3_Vision

- Verwendung von selbstgeschriebenen TwinCAT SPS oder C++-Algorithmen
- Verwendung der TwinCAT Vision API in einem C++ Modul
- Verwendung von Matlab/Simulink
- Abrufen der Bilddaten per ADS aus anderen Applikationen

Was ist der Unterschied zwischen Bildverarbeitungssoftware wie TwinCAT Vision und Smart-Kameras?

Bildverarbeitungssoftware wie TwinCAT Vision:

- Empfangen Bilddaten von einer Kamera
- Verarbeiten diese auf einem Computer
- Senden verarbeitete Informationen an die Steuerung
- Vorteil von TwinCAT Vision: Steuerung und Bildverarbeitung ist dieselbe Software und laufen im selben System. Daher entfallen zusätzliche Schnittstellen und Protokolle.

Smart-Kameras:

- Verfügen über internen Prozessor
- Führen Bildverarbeitungsoperationen direkt auf der Kamera aus
- Liefern verarbeitete Informationen wie z. B. Gut- / Schlechtinformationen oder die Position eines Werkstücks über zusätzliche Schnittstellen

Kann TwinCAT Vision auf einer GPU ausgeführt werden?

Nein, TwinCAT Vision läuft ausschließlich in der TwinCAT Echtzeitumgebung auf einer CPU. Alternativ bietet TwinCAT Vision die Möglichkeit einer automatisch parallelisierten Ausführung, indem die Multi-Core Funktionalität von TwinCAT (in Zusammenspiel mit Job Tasks [▶ 66]) genutzt wird.

9.2 Wichtige Pfade

Pfade für Vision-Daten

HINWEIS

Datenrelevante Pfade

Stellen Sie sicher, dass folgende Standard-Pfade nicht durch Write-Filter oder ähnliche Programme beeinträchtigt werden. Eine Nichtbeachtung kann zu Datenverlust führen. Wenn die entsprechenden Pfad-Einstellungen manuell angepasst wurden, gilt gleiches für die eingestellten Pfade.

Verwendung	Pfad
Speicherort für File Source Bilder	C:\Users\Public\TcVision\FileSources\<UniqueId> (Wird automatisch vom File Source festgelegt. <UniqueId> ist für jedes Projekt unterschiedlich.)
Speicherort für Kamera-Streams	C:\Users\Public\TcVision\CameraStreams (Kann je nach Einstellung in Record/Playback [▶ 109] abweichen.)
Speicherort für Bilder, die durch SPS-Funktionen gelesen oder geschrieben werden.	C:\Users\Public\TcVision\Images (Kann je nach Einstellung in Service Configuration [▶ 68] abweichen.)
Speicherort für Container, die durch SPS-Funktionen gelesen oder geschrieben werden.	C:\Users\Public\TcVision\Containers (Kann je nach Einstellung in Service Configuration [▶ 68] abweichen.)
Speicherort für ML-Modelle, die durch SPS-Funktionen gelesen oder geschrieben werden.	C:\Users\Public\TcVision\MlModels (Kann je nach Einstellung in Service Configuration [▶ 68] abweichen.)
Speicherort für Kalibrierergebnisse, die durch SPS-Funktionen gelesen oder geschrieben werden.	C:\Users\Public\TcVision\CalibrationResults (Kann je nach Einstellung in Service Configuration [▶ 68] abweichen.)
Speicherort für individuelle Kalibriermuster	C:\Users\Public\TcVision\CalibrationPattern

Pfade für Log-Dateien

Log-Datei	Pfad
TwinCAT Vision Installations-Log (alle anderen TwinCAT Installations-Logs liegen auch in diesem Ordner)	C:\Users\<*User*>\ TF7xxx-Vision.exe.log Ab Build 4026: C:\ProgramData\Beckhoff\TcPkg\logs\ Die individuellen Produkt Log-Dateien befinden sich im Unterordner MSI.
TwinCAT Vision Service Log	C:\ProgramData\Beckhoff\TcVnService\TcVnService.log
Allgemeiner TwinCAT Vision Log	C:\ProgramData\Beckhoff\Vision (Kann je nach Einstellung in Logging [▶ 71] abweichen)
Ergebnisbilder des Kalibrationsassistenten [▶ 116] zur Analyse der Erkennung von Merkmalspunkten auf dem Kalibrationsmuster	C:\ProgramData\Beckhoff\Vision_CalibrationAssistantOutput

9.3 Troubleshooting

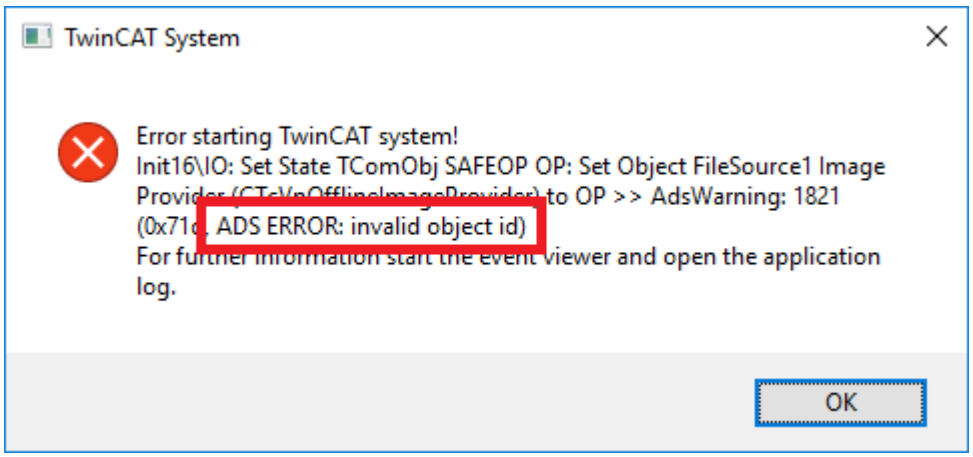
Bekannte Fehlersituationen und entsprechende Lösungen sind folgend beschrieben. Wenn Sie ein Problem konkret auf eine Funktion oder einen Funktionsblock zurückführen können, lesen Sie zunächst den entsprechenden Eintrag in der API-Referenz [▶ 146]!

9.3.1 TwinCAT System Start

Folgend sind Lösungen für Situationen erklärt, in denen der Start des TwinCAT Systems verhindert wird.

9.3.1.1 TwinCAT System startet nicht

Wenn das TwinCAT System nicht gestartet werden kann, erscheint häufig folgende Meldung. Anhand des rot-umrandeten Fehlercodes lassen sich Hinweise zum Beheben des Problems geben. Im Weiteren finden Sie Hinweise zu den üblichen Fehlercodes bei TwinCAT Vision Applikationen.



Invalid Object ID

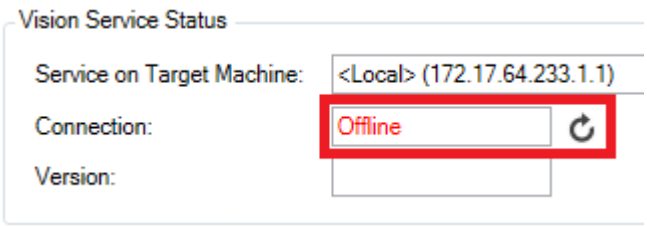
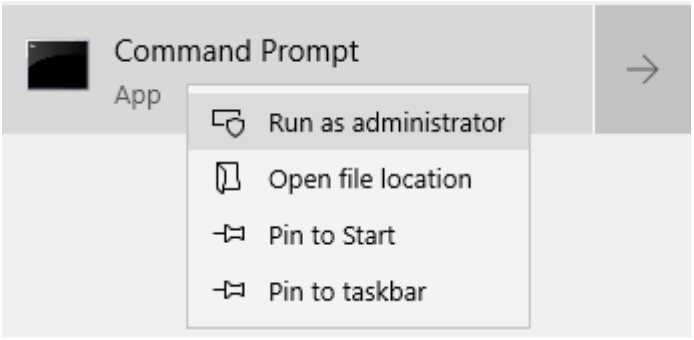
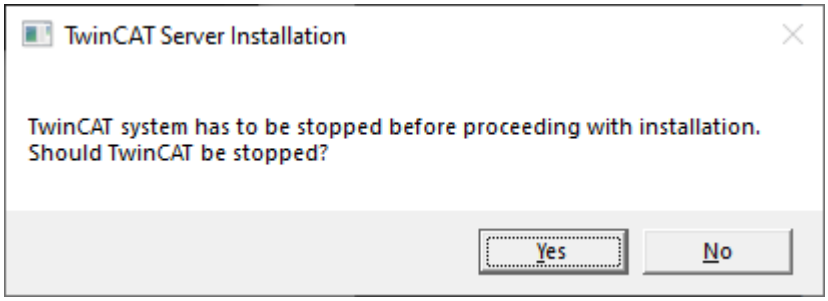
Fehlermeldung	ADS ERROR: invalid object id
Lösung 1	**Keine Task verknüpft** 1. Überprüfen Sie, ob für jedes verwendete Vision-Gerät die entsprechenden Tasks angelegt sind und ergänzen Sie fehlende Tasks. ◦ Überprüfen Sie, ob für jedes File Source Control eine Task für die Image Acquisition angelegt ist (Standardzykluszeit 10 ms). ◦ Überprüfen Sie, ob für jede GigE Vision Kamera eine Task für die Image Acquisition (Standardzykluszeit 1 ms) und - wenn ein Ads Communicator vorhanden ist - auch für diesen eine Task angelegt ist (Standardzykluszeit 10 ms). 2. Die Tasks müssen den entsprechenden TcCom-Modulen der Vision Geräte zugeordnet sein. Überprüfen Sie die Zuordnung und korrigieren Sie diese bei Bedarf. ◦ Überprüfen Sie, ob die Image Provider Task des File Source mit dem Image Provider des File Source Controls verknüpft ist. ◦ Überprüfen Sie, ob die Image Acquisition Task der Kamera mit der Image Acquisition und der Image Acquisition Simulation der Kamera verknüpft ist. ◦ Verknüpfen Sie die Ads Communicator Task mit dem Ads Communicator der Kamera. ◦ Weisen Sie dem Image Provider der Kamera keine Task zu.
Lösung 2	**Netzwerkadapter ungültig** Der Netzwerkadapter wurde z. B. bei einem Systemwechsel nicht angepasst. Passen Sie den Netzwerkadapter an das aktuelle Target-System an. Öffnen Sie dazu im entsprechenden Netzwerkadapter das Tab **Adapter** und wählen Sie mit **Search…** einen neuen kompatiblen Adapter aus.
Lösung 3	**Netzwerkadapter gelöscht** Der Netzwerkadapter wurde z. B. auf die Vermutung hin gelöscht, dass er bei der Simulation der Kamera nicht benötigt würde. Jedoch benötigt jedes Kameraobjekt immer einen Netzwerkadapter; bei Bedarf kann dieser deaktiviert, aber nicht gelöscht werden. Erstellen Sie einen neuen Netzwerkadapter mit einem IpStack und verknüpfen Sie den IpStack im Tab **Interface Pointer** des Acquisition-Objekts der Kamera.

Weitere Informationen zu der Verknüpfung von Vision TcCOM-Objekten finden Sie in den entsprechenden Kapiteln der Kamera [▶ 132] und des File Source [▶ 139].

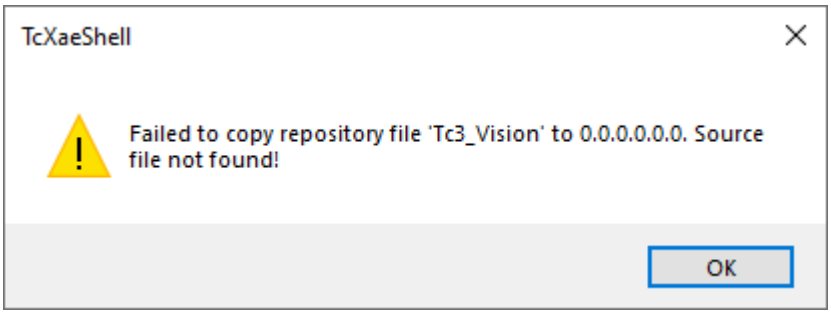
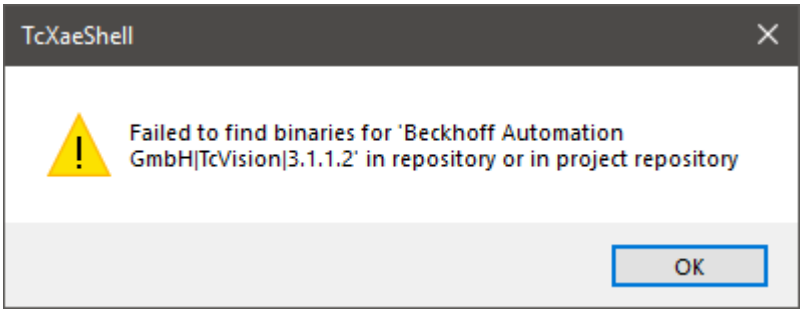
General ADS Error

Fehlermeldung	ADS ERROR: General ADS Error
Lösung 1	**Mehr Kameras als konfiguriert** Die Standard-Einstellung des Netzwerkadapters erlaubt es, bis zu 2 Kameras gleichzeitig zu betreiben. Wenn Sie mehr Kameras an demselben Adapter betreiben möchten, erhöhen Sie die Init-Parameter `ipMaxReceiver` und `UdpMaxReceiver` auf die doppelte Anzahl der verbundenen Kameras (siehe Konfiguration der Netzwerkadapter [▶ 58]).

9.3.1.2 Vision Service startet nicht

Situation	Der TwinCAT Vision Service startet nicht. Dies ist im VISION-Knoten unter dem Tab **Service Konfiguration** an dem Connection Status „Offline“ zu erkennen. 
Lösung	Wenn der TwinCAT Version Service nur einmalig nicht startet oder ausfällt, kann dieser mit einem TwinCAT Neustart (Klick auf) reaktiviert werden. Wenn er jedoch grundsätzlich (nach dem Systemstart) nicht startet, muss der Service neu als TwinCAT Service registriert werden. Folgen Sie dazu diesen Schritten: 1. Schalten Sie TwinCAT in den Konfig-Modus 2. Öffnen Sie eine Eingabeaufforderung als Administrator.  3. Wechseln Sie per „`cd C:\TwinCAT\3.1\Components\Vision\TcVision`” in den entsprechenden Ordner. 4. Führen Sie den Befehl „`TcVnService.exe /tc3serverreg`” zum Starten des Vision Service aus. 5. Bestätigen Sie den folgenden Dialog mit einem Klick auf **Ja**.  ⇨ Der Vision Service wird gestartet. **TwinCAT muss sich im Konfig-Modus befinden, damit der Vision Service gestartet werden kann. Daher ist ein Neustart der Maschine notwendig, wenn der Vision Service neu gestartet werden soll.** Falls der Vision Service manuell gestoppt werden soll, können Sie den Befehl „`TcVnService.exe /unregtc3server`“ verwenden.

9.3.1.3 Binary oder Repository konnte nicht gefunden / geladen werden

Situation	Eine der beiden Meldungen erscheint beim Aktivieren der Konfiguration. Dabei können, je nach Ursache und installierter Version, andere Dateinamen oder Versionsnummern in der Meldung vorkommen. Die Ursache für die Meldung ist, dass die im Projekt verwendete Version nicht mit der auf dem System installierten Version übereinstimmt.  
Lösung 1	**Überprüfen der Vision Bibliothek** Überprüfen Sie die Version der im Projekt eingebundenen Tc3_Vision Bibliothek. Wenn die Version nicht mit der im Ordner C:\TwinCAT\3.1\Repository\Beckhoff Automation GmbH\Tc3_Vision des Entwicklungsrechners übereinstimmt, aktualisieren Sie entweder die Bibliothek im Projekt oder installieren Sie eine andere Setup-Version.
Lösung 2	**Überprüfen der TMC-Version von Kamera oder FileSorce** Überprüfen Sie unter **SYSTEM** > **TcCOM Objects** > **Project Objects** die im Projekt verwendete TMC-Version der aufgelisteten Objekte. Ist die Versionsangabe gelb hinterlegt ist die angegebene Version auf dem System nicht vorhanden oder stimmt nicht überein. Aktualisieren Sie nun entweder die entsprechenden TMC-Versionen (Auswahl über Pull-Down der Version, dann Rechtsklick auf eine der Objekt-IDs in der ersten Spalte und wählen Sie **Reload TMI/TMC Description(s) with changed version**) auf die verfügbare Version oder installieren Sie eine andere Setup-Version.
Lösung 3	**Die Vision Komponente ist nicht vorhanden** Die entsprechende Repository-Datei fehlt auf dem aktuellen System, weil die Vision Komponente bei der Installation nicht ausgewählt wurde. Grundsätzlich müssen die Versionen des Entwicklungs- und Target-Systems übereinstimmen. Kontrollieren Sie bei unterschiedlichen Systemen daher beide Installationen. Auf reinen Runtime-Systemen XAR kann nur der Vision Service installiert werden, da alle anderen Dateien vom Entwicklungsrechner XAE übertragen werden. Installieren Sie die fehlende Vision Komponente oder führen Sie das Setup mit der Reparatur-Option aus.

9.3.2 SPS-Laufzeitumgebung

Folgend sind Lösungen für Situationen erklärt, in denen Fehler während der Ausführung der SPS auftreten.

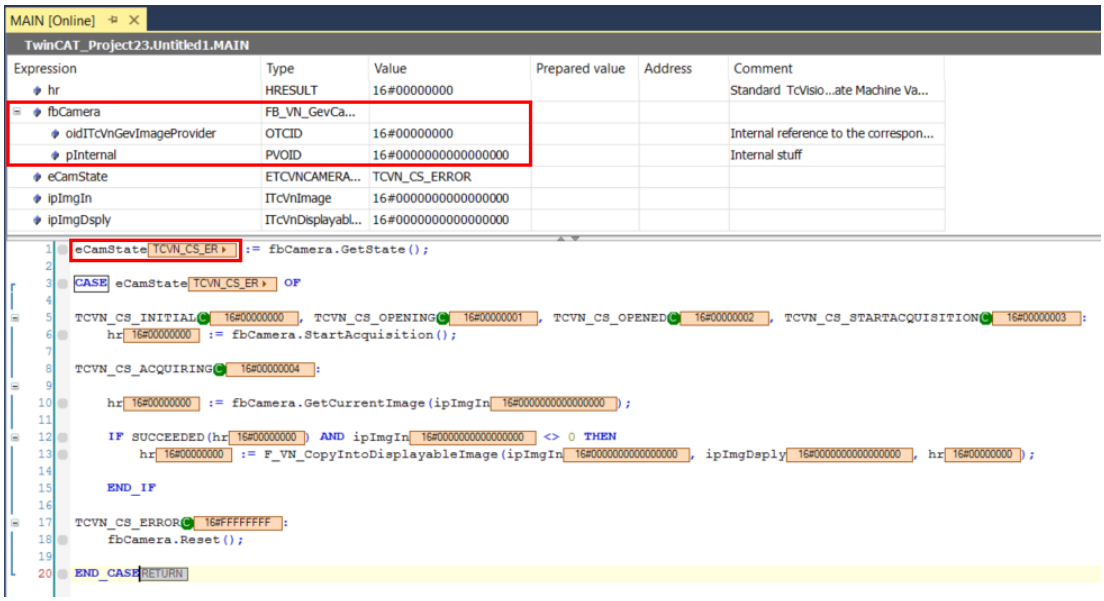
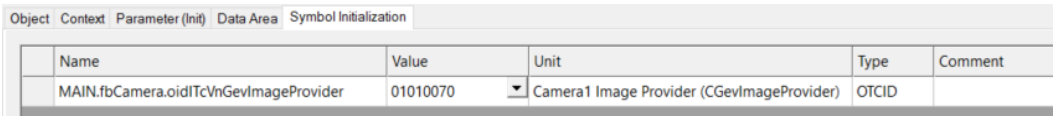
9.3.2.1 Vision-Gerät im ERROR-Zustand

Kamerafunktionsblöcke wechseln zwischen ERROR- und INITIAL-Zustand

Situation	Die Instanz des Funktionsblocks FB_VN_Gev_CameraControl [▶ 1672] bzw. FB_VN_SimpleCameraControl [▶ 1697] gibt beim Aufruf der Methode `GetState` abwechselnd den `TCVN_CS_ERROR` und `TCVN_CS_INITIAL` zurück.
	Im Zustand `TCVN_CS_ERROR` wird erfolgreich die Reset() Methode des Funktionsbausteins ausgeführt und die Kamerainstanz wechselt in den Zustand `TCVN_CS_INITIAL`. Bleibt die Kamera im `TCVN_CS_ERROR`-Zustand, sehen Sie nach unter Gerät dauerhaft im ERROR-Zustand [▶ 2954].
	`eState := fbCamera.GetState()`
	`IF eState = TCVN_CS_ERROR THEN`
	`  fbCamera.Reset();`
	`END_IF`
	
	Im Zustand TCVN_CS_INITIAL wird entweder die Funktionsbausteinmethode OpenCamera() oder StartAcquisition() aufgerufen, wobei der Aufruf fehlschlägt und die Instanz zurück in den Zustand TCVN_CS_ERROR geht.
	`eState := fbCamera.GetState()`
	`IF eState = TCVN_CS_INITIAL THEN`
	`  fbCamera.OpenCamera();`
	`END_IF`
	bzw.
	`eState := fbCamera.GetState()`
	`IF eState = TCVN_CS_INITIAL THEN`
	`  fbCamera.StartAcquisition();`
	`END_IF`

Lösung 1	Prüfen Sie die physikalische Verbindung zur Kamera und die Spannungsversorgung der Kamera.
Lösung 2	Prüfen Sie die Einstellungen zur Kameraverbindung (IP-Adresse der Kamera, Netzwerkport).

Funktionsblock dauerhaft im ERROR-Zustand

Situation	Die Instanz des Funktionsbausteins `FB_VN_GevCameraControl`, `FB_VN_SimpleCameraControl` oder `FB_VN_FileSourceControl` gibt dauerhaft den Zustand `TCVN_CS_ERROR` zurück. Auch beim Aufruf der Funktionsbaustein-Methode Reset erfolgt kein Zustandswechsel in den `TCVN_CS_INITIAL` Zustand (falls doch, siehe Gerät wechselt zwischen ERROR- und INITIAL-State [▶ 2953]). `fbCamera.Reset();` 
Lösung 1	Die Symbol-Instanz muss mit dem Image Provider verknüpft sein - jede Instanz eines `FB_VN_GevCameraControl`, eines `FB_VN_SimpleCameraControl` und eines `FB_VN_FileSourceControl` muss jeweils mit einer im Vision-Knoten angelegten Kamera bzw. eines angelegten File Source Controls verknüpft werden. Gehen Sie dazu folgendermaßen vor: 1. Führen Sie ein erfolgreiches Rebuild des SPS-Projektes durch. 2. Doppelklicken Sie anschließend auf die Instanz des SPS-Projektes. 3. Weisen Sie dort unter dem Tab **Symbol Initialization** den Instanzen von `FB_VN_GevCameraControl` jeweils den Image Provider der dazugehörigen Kamera, des dazugehörigen File Source Controls zu. 4. Aktivieren Sie die Konfiguration erneut.  **Der FB_VN_GevCameraControl kann nur mit dem Image Provider einer Kamera verknüpft werden.** **Der FB_VN_FileSourceControl kann nur mit dem Image Provider eines File Source Controls verknüpft werden.** **Der FB_VN_SimpleCameraControl kann sowohl mit dem Image Provider einer Kamera als auch mit dem Image Provider eines File Source Controls verknüpft werden.**
Lösung 2	Der Instanz von `FB_VN_GevCameraControl` bzw. `FB_VN_SimpleCameraControl` bzw. `FB_VN_FileSourceControl` ist der Image Provider einer Kamera bzw. eines File Source Controls zugewiesen, dennoch ist das Gerät dauerhaft im Error-Zustand. Prüfen Sie die Methoden-Aufrufe der Instanz von `FB_VN_GevCameraControl` bzw. `FB_VN_SimpleCameraControl` bzw. `FB_VN_FileSourceControl`. Viele Methoden dürfen nur in einem bestimmten Zustand aufgerufen werden. Eine detaillierte Beschreibung der Zustände von Vision-Geräten finden Sie in der Sektion Funktionsblöcke zur Bildaufnahme [▶ 1662].

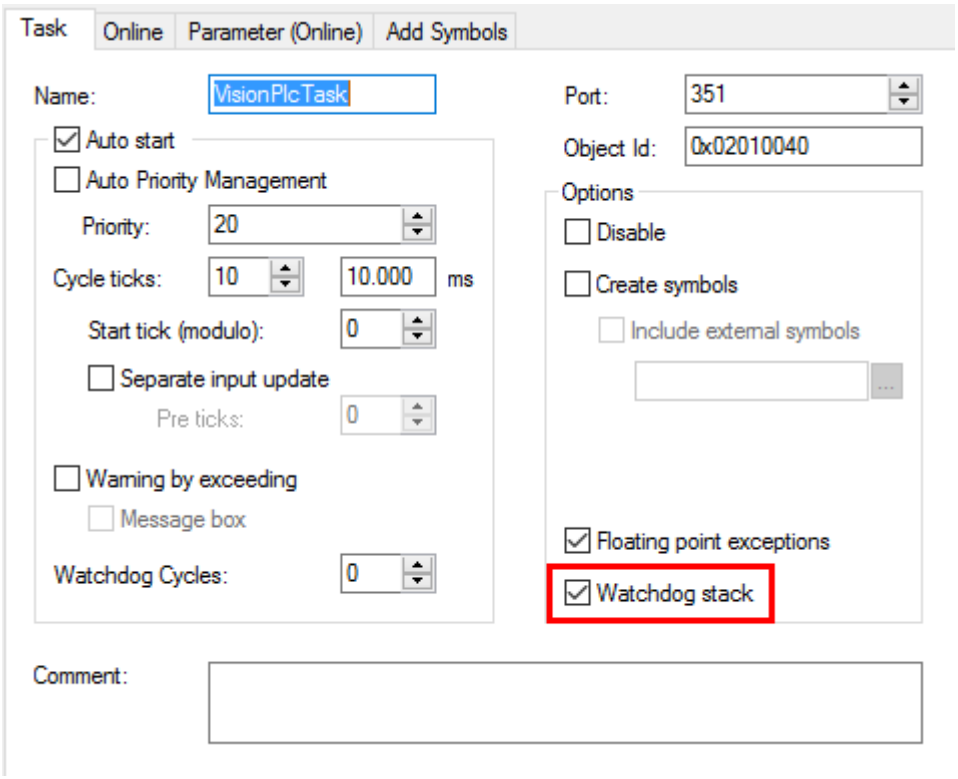
	• In allen Zuständen ◦ `GetState` ◦ `GetCalibPatternRef` (von FB_VN_GevCameraControl) ◦ `GetCameraMatrix` (von FB_VN_GevCameraControl) ◦ `GetDistortionCoefficients` (von FB_VN_GevCameraControl) ◦ `GetRotationMatrix` (von FB_VN_GevCameraControl) ◦ `GetTranslationVector` (von FB_VN_GevCameraControl) ◦ `SetCameraMatrix` (von FB_VN_GevCameraControl) ◦ `SetDistortionCoefficients` (von FB_VN_GevCameraControl) ◦ `SetRotationMatrix` (von FB_VN_GevCameraControl) ◦ `SetTranslationVector` (von FB_VN_GevCameraControl) • Im INITIAL-Zustand und im INITIALIZING-Zustand ◦ `InitializeCamera` (von FB_VN_GevCameraControl) • Im INITIAL-Zustand und im OPENING-Zustand ◦ `OpenCamera` (von FB_VN_GevCameraControl) ◦ `StartAcquisition` • Im OPENED-Zustand und im STARTACQUISITION-Zustand ◦ `StartAcquisition` • Im OPENED-Zustand und im CLOSING-Zustand ◦ `CloseCamera` (von FB_VN_GevCameraControl) • Im ACQUIRING-Zustand ◦ `GetCurrentImage` ◦ `GetCurrentImageUndistorted` (von FB_VN_GevCameraControl) ◦ `GetCurrentImageWithGvspInfo` (von FB_VN_GevCameraControl) ◦ `GetCurrentImageAndFileName` (von FB_VN_FileSourceControl) ◦ `ClearImageQueue` (von FB_VN_GevCameraControl) ◦ `GetLastImageFromQueue` (von FB_VN_GevCameraControl) ◦ `GetOmittedImagesNum` (von FB_VN_GevCameraControl) • Im ACQUIRING-Zustand und im TRIGGERING-Zustand ◦ `TriggerImage` ◦ `TriggerImageExp` (von FB_VN_FileSourceControl) ◦ `TriggerImageByName` (von FB_VN_FileSourceControl) • Im ACQUIRING-Zustand und im STOPACQUISITION-Zustand ◦ `StopAcquisition` • Im ERROR-Zustand ◦ `Reset`

9.3.2.2 Kamera nach Neustart nicht gefunden

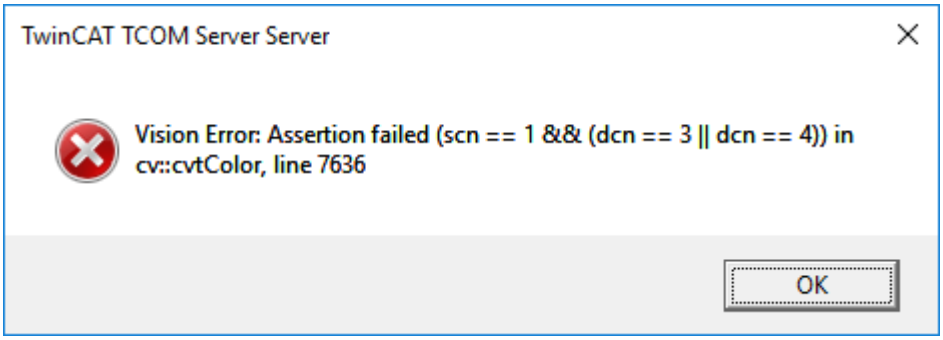
Situation	Applikation funktioniert ganz normal. Nach Neustart der Maschine kann keine Verbindung zur Kamera hergestellt werden.
Lösung 1	Die IP-Adresse hat sich geändert. Nutzen Sie die Force-IP-Option im Konfigurationsassistenten [▶ 79] der Kamera.

9.3.2.3 Fehler beim Starten eines Watchdogs

Situation	Die Funktion F_VN_StartAbsWatchdog [▶ 1020] bzw. F_VN_StartRelWatchdog [▶ 1022] gibt dauerhaft den Fehlercode 16#71A (NOINTERFACE) zurück.
	
	`hrWD 16#9811071A := F_VN_StartRelWatchdog(5000, S_OK);`
	`hr 16#9811070C := F_VN_DetectBlobs(ipImgIn 16#FFFF900DA079D3`
	`hrWD 16#9811071A := F_VN_StopWatchdog(hrWD 16#9811071A, tRest`

Lösung	Aktivieren Sie den Watchdog Stack in der dazugehörigen SPS-Task und aktivieren Sie anschließend die Konfiguration erneut. 

9.3.2.4 Vision Error: Assertion failed

Fehlermeldung	Während der Ausführung einer Vision-Applikation erscheint eine Fehlermeldung, ähnlich wie die folgende. TwinCAT geht dabei in den EXCEPTION-Zustand.  Dabei handelt es sich um die Meldung, dass ein Konflikt in der Ausführung einer Vision-Funktion vorliegt, der nicht abgefangen werden konnte. Dies bezieht sich im Wesentlichen auf das korrekte Format der Eingangsdaten und sollte nur auftreten, wenn die Funktion nicht in Übereinstimmung mit der API-Referenz benutzt wird. Die Fehlerspezifikation hinter „VisionError: Assertion failed“ kann je nach Anwendungsfall abweichen.
Lösung bei F_VN_ConvertColorSpace [▶ 1270]	Ein Bild kann sowohl 1-kanalig als auch mehrkanalig sein. Bei bestimmten Funktionsaufrufen wird ein Bild mit einer definierten Kanalanzahl erwartet. Zum Beispiel bei der Funktion `F_VN_ConvertColorSpace`, wo über das Enum `ETcVnColorSpaceTransform` angegeben wird, welches Format das Eingangsbild hat. Bei eingeloggtem Engineering wird der entsprechende Funktionsaufruf im SPS-Code markiert. Bei der Analyse, warum die tatsächliche Kanalanzahl von der erwarteten abweicht, kann die Funktion `F_VN_GetPixelFormat` helfen, die die tatsächliche Kanalanzahl zurückgibt. Typische Ursachen: • File Source Control Im File Source ist ein anderes Bildformat eingestellt als erwartet wird oder das Bildformat steht auf „Original Format from File“ und das tatsächliche Bildformat ist ein anderes als erwartet. So sind zum Beispiel monochrom aussehende Bilder nicht grundsätzlich als monochrome Bilder (1-Kanal-Bild) gespeichert. • Kamera Bei vielen Kameras kann in der Konfiguration das Pixelformat eingestellt werden. Abhängig vom ausgewählten Format verändert sich auch die Bild-Kanalanzahl. • Vorherige Verarbeitungsschritte Durch einen vorherigen Funktionsaufruf kann sich das Bild und dadurch auch die Kanalanzahl verändert haben.

9.3.2.5 Error: SSE Invalid Operation

Fehlermeldung	Sie bekommen eines Fehlermeldung namens **SSE Invalid Operation**.
Lösung	Dieser Fehler kann unnötigerweise auftreten, wenn das **Floating point exceptions** Handling auf der ausführenden Task aktiviert ist. Befolgen Sie die Hinweise in der Sektion CPU-Kerne und Tasks [▶ 63], indem Sie die Option **Floating point exceptions** deaktivieren.

9.3.2.6 File Source

Fehlerzustand

Fehlerbild	Das File Source wechselt bei laufender SPS dauernd in den Fehlerzustand.
Lösung 1	**Zu wenig Speicher** Der Router-Speicher ist zu klein eingestellt. Daher kann das File Source kein Bild in die SPS laden und geht bei dem Versuch in den Fehlerzustand. Vergrößern Sie den Router-Speicher.

Bilder aus FS Control nicht in SPS

Fehlerbild	Im File Source Control sind Bilder ausgewählt und der File Source Funktionsblock in der SPS befindet sich im `ACQUISITION` Zustand. Dennoch können über `GetCurrentImage` keine Bilder abgerufen werden.
Lösung 1	**BaseDir nicht gesetzt** Der Button **Read from Target** wurde erstmals betätigt, nachdem TwinCAT aktiviert wurde. In diesem Fall ist der Pfad, aus dem die File Source Bilder ausgelesen werden noch nicht an das laufende TcCOM-Objekt übergeben. Daher können keine Bilder eingeladen werden, obwohl Sie im File Source Control ausgewählt sind. Aktivieren Sie TwinCAT erneut, nachdem Sie **Read from Target** gedrückt haben.
Lösung 2	**Trigger Modus** Im File Source Control ist der Trigger-Modus ausgewählt, aber in der SPS wird nur die Methode `GetCurrentImage`, nicht `TriggerImage` aufgerufen. Triggern Sie ein Bild durch Aufruf der Methode `TriggerImage` in der SPS oder deaktivieren Sie den Trigger Modus im File Source Control.

9.3.3 Assistenten

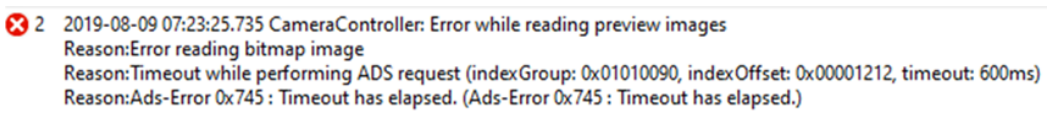
Folgend sind Lösungen für Situationen erklärt, in denen Probleme bei der Bedienung von Assistenten auftreten.

9.3.3.1 Timeouts

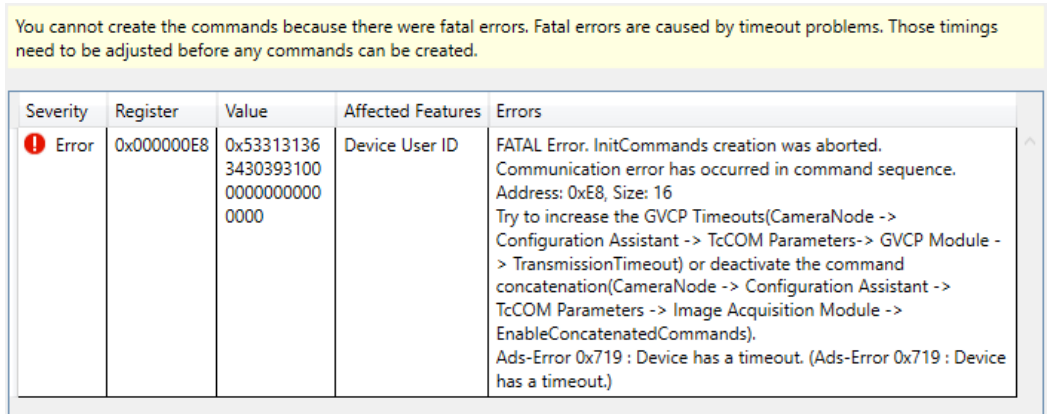
Situation

Ein Timeout-Fehler tritt während der Kommunikation mit einer Kamera auf. Dies kann an unterschiedlichen Stellen passieren wie die folgenden Fälle beispielhaft zeigen.

Bei der Bildanzeige in der Entwicklungsumgebung:

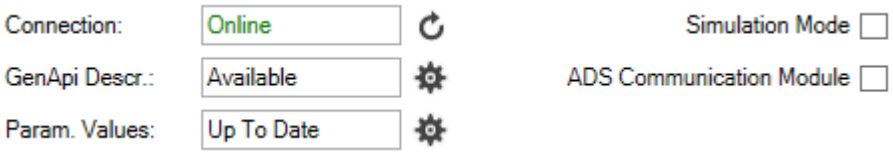


Beim Erstellen der Init-Commands:



Bei Verwendung des ForceIP-Befehls:

Beim Verbindungsaufbau im Camera-Objekt:



Lösung

Im Fall von Timeout-Fehlern sind im Wesentlichen zwei verschiedene Timeout-Arten von Relevanz: Zum einen der Transmission-Timeout, der die Kommunikation zwischen dem Image Acquisition Modul und der Kamera limitiert. Zum anderen der ADS-Communication-Timeout, der die Kommunikation zwischen der Benutzeroberfläche und dem Image Acquisition Modul limitiert.

Passen Sie den Transmission-Timeout in den TcCOM-Parametern des GVCP Moduls im Konfiguration [▶ 83] an. Passen Sie ggfs. ebenfalls die Anzahl MaxTimeouts an.

Achten Sie darauf, dass Sie bei Bedarf ebenfalls den ADS-Communication-Timeout in den Einstellungen [▶ 91] anpassen, da dieses größer als das Produkt aus MaxTimeouts und Transmission-Timeout sein sollte.

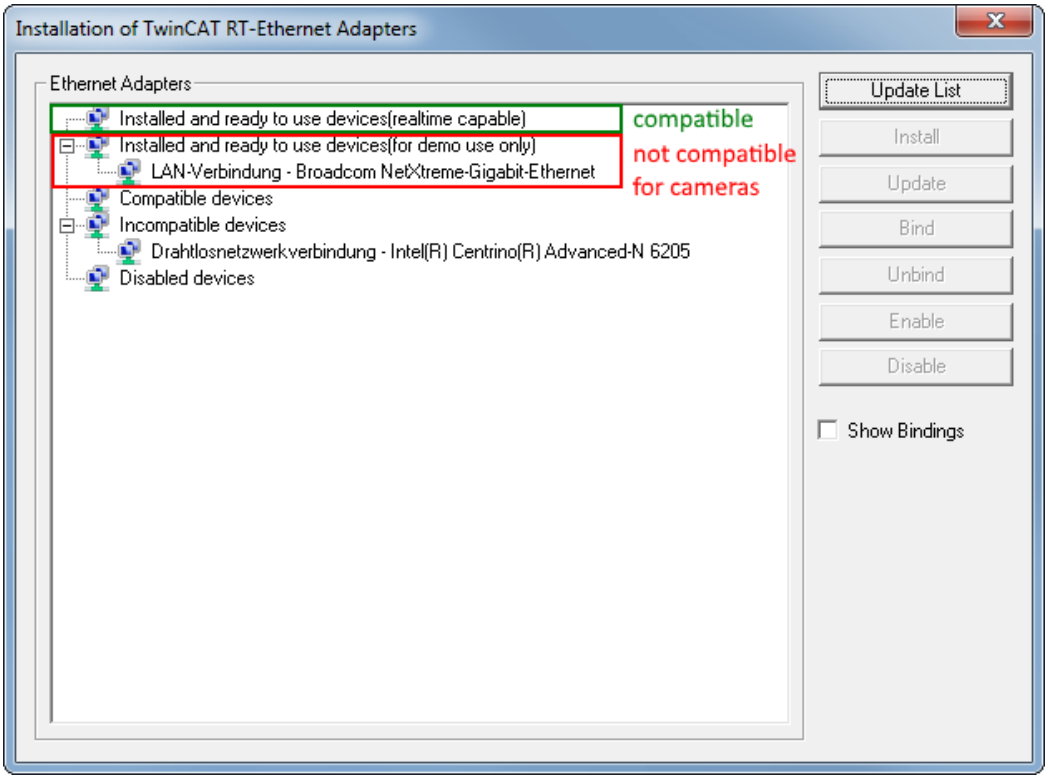
Erhöhen Sie schrittweise die Zeiten und ggf. die Anzahl der Versuche so lange bis keine Fehlermeldung mehr auftritt. Achten Sie aber darauf, dass Sie die Zeiten und Anzahl der Versuche nicht unnötig hoch wählen, da sonst sehr lange Wartezeiten entstehen können.

9.3.3.2 Stockende Bildanzeige

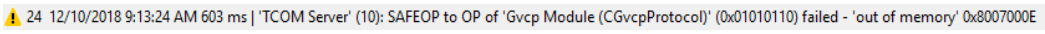
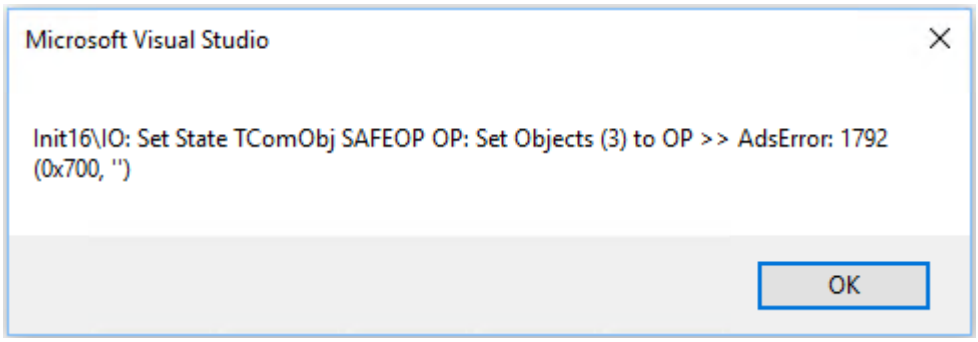
Fehlerbild	Im ADS Image Watch und Kameraassistenten bleibt die Bildanzeige zwischenzeitlich stehen.
Lösung 1	**Remote Desktop Protocol** In Einzelfällen kann es vorkommen, dass das Remote Desktop Protocol und die Netzwerkkarte nicht gut mit der Übertragung von der TwinCAT Vision Entwicklungsumgebung bzw. den Bildanzeigetools funktionieren. Arbeiten Sie entweder lokal und verbinden sich in TwinCAT mit dem Target oder arbeiten Sie direkt auf dem Zielrechner.

Fehlerbild	Im ADS Image Watch und Kameraassistenten werden nicht alle Bilder angezeigt.
Lösung 1	**Reduzieren der Datenmenge** Da zum Anzeigen der Bilder diese per ADS aus der Steuerung abgerufen werden, kann es gerade bei großen Bildern, hohen Bildfrequenzen oder mehreren ADS Image Watch-Fenstern dazu kommen, dass nicht alle Bilddaten abgerufen werden können. Reduzieren Sie daher die Anzahl an gleichzeitig anzuzeigenden Bildern, reduzieren Sie die Bildfrequenz oder die Datenmenge pro Bild durch Verwendung von ROIs.

9.3.3.3 Kamera bei Device Discovery nicht gefunden

Situation	Sie sind dabei, ein GigE Vision Kamera Objekt anzulegen. Sie haben erfolgreich einen TwinCAT RT-Ethernet Adapter mit IP-Stack eingerichtet. Dennoch bleibt der Device Discovery Dialog leer und es gibt keine Fehlermeldung.
Lösung 1	**Falscher Netzwerkadapter ausgewählt**
Lösung 2	**Keine GigE Vision Kamera** Eine Kamera mit Gigabit Ethernet Schnittstelle muss nicht zwingend GigE Vision sprechen. Achten Sie bitte genau auf die Herstellerangabe zum Übertragungsprotokoll.
Lösung 3	**Inkompatibler Netzwerkadapter** Der verwendete Netzwerkadapter... 

9.3.3.4 Fehler beim Laden von Kamera-Objekten

Fehlermeldung	 Im Konfigurationsmodus erscheint folgende Meldung von Visual Studio, wenn die Kamera-Module neu geladen werden oder eine Kamera über den Discover Devices Button im Choose Target Camera Dialog ausgewählt werden soll. 
Lösung	Es sind mehr Kameras eingebunden als konfiguriert wurden. Die Standard-Einstellung des Netzwerkadapters erlaubt es, bis zu 2 Kameras gleichzeitig zu betreiben. Wenn Sie mehr Kameras betreiben möchten, erhöhen Sie die Init-Parameter `ipMaxReceiver` und `UdpMaxReceiver` des Netzwerkadapters auf die doppelte Anzahl der verbundenen Kameras (siehe Konfiguration der Netzwerkadapter [▶ 62]).

9.3.3.5 Warnung: Reading/Writing is not permitted

Warnung	 Diese Warnung kann beim Verbinden einer Kamera erscheinen.
Lösung	Diese Warnung zeigt an, dass ein Schreib- oder Lese-Befehl, den TwinCAT Vision an die Kamera sendet, nicht erlaubt ist. Dies kann passieren, wenn die Definition der Zugriffsrechte auf ein Register in der GenAPI-Beschreibung von dem tatsächlichen Verhalten der Kamera abweicht. Im Regelfall handelt es sich dabei um für den Anwender unwichtige Kameraregister. Um sicher zu gehen, welche Register dies betrifft, können Sie im Konfigurationsassistenten [▶ 79] nach der angezeigten Registeradresse suchen.

9.3.4 TwinCAT HMI Pakete

Folgend sind Lösungen für Situationen erklärt, in denen TwinCAT HMI Pakete nicht wie erwartet funktionieren.

9.3.4.1 Server-Erweiterung

Bilder werden nicht angezeigt

Situation	Bilder aus der SPS werden nicht in der HMI angezeigt, obwohl die Server-Erweiterung [▶ 2784] installiert und die Bild-Symbole passend verknüpft sind.
Lösung 1	**Fehlende Administratorrechte** Sie haben als Bildformat „Stream“ ausgewählt, den HMI-Server jedoch nicht als Administrator gestartet. Wählen Sie entweder ein anderes Bildformat (BMP, PNG oder JPG) oder starten Sie den HMI-Server als Administrator.

Bilder fehlen auf gepublishter Seite

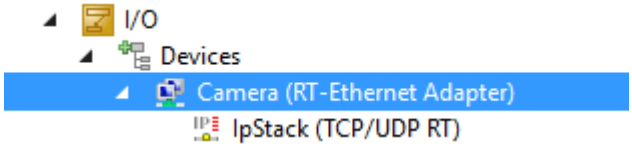
Situation	Die TcHmiVision [▶ 2784] Server-Extension ist erfolgreich ins Projekt eingebunden. Im Live View bzw. auf dem vom Engineering-Server bereitgestellten HMI werden Bilder angezeigt. Das Projekt konnte auch fehlerfrei auf dem Standalone-TcHmi-Server gepublished werden. Jedoch werden auf dem gepublishten HMI keine Bilder angezeigt.
Lösung 1	**Fehlende Target-Lizenzen** In der TC3 HMI Server-Lizenz ist eine Target-Lizenz enthalten. Soll der HMI-Server mit mehreren Targets (ADS-Verbindungen) kommunizieren, so werden weitere Lizenzen benötigt.
Lösung 2	**Falsche Konfiguration der Laufzeitumgebung** Die Default-Konfiguration, die im Engineering verwendet wird, kann von der gepublishten Konfiguration abweichen (unterschiedliche AmsNetIds). Daher überprüfen Sie die Konfiguration der Laufzeitumgebung und passen Sie sie gegebenenfalls an.
Lösung 3	**Eingeschränkte Benutzerrechte** Die Benutzerverwaltung des TwinCAT HMI wird verwendet und der aktuell eingeloggte Benutzer hat keine Berechtigungen, die Bilder zu sehen oder die Daten der Extension zu empfangen. Überprüfen Sie die Berechtigungen Ihrer Benutzergruppen.
Lösung 4	**Begrenzen der Datenmenge oder des Aktualisierungintervalls** Da die Datenmenge insbesondere bei mehreren Bildern recht groß werden kann, die gleichzeitig per ADS aus der Steuerung abgerufen und vom HMI-Server konvertiert werden muss, kann dies zu Engpässen führen. Zur Abhilfe stehen mehrere Möglichkeiten zur Verfügung, Verkleinern der Bilder in der Steuerung, Erhöhen der Aktualisierungsraten der einzelnen Bilder, Verwenden verschiedener Aktualisierungsraten oder die Verwendung eines komprimierten Bildformats.

9.3.5 Kamerakommunikation

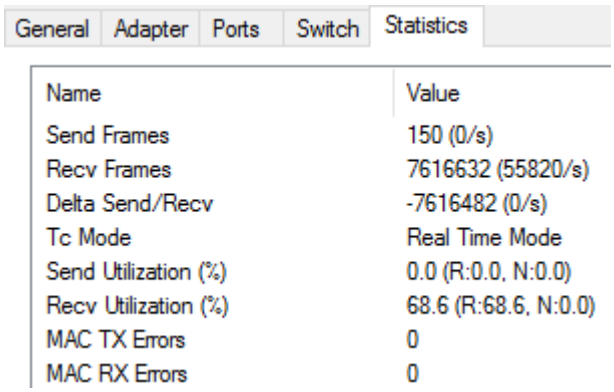
Bei Problemen mit der Verbindung, speziell mit gesendeten oder empfangenen Frames, sowie Problemen bei der Parametrierung oder fehlenden Bildern finden Sie im Netzwerk-Adapter und unter dem Image Acquisition Module des Kamera-Objekts folgende Diagnoseinformationen.

Diagnosemöglichkeiten im RT-Ethernet Adapter

1. Wählen Sie zuerst den für die jeweilige Kamerakommunikation verwendeten RT-Ethernet Adapter aus.



2. Öffnen Sie diesen per Doppelklick und gehen zu dem **Statistics** Tab.

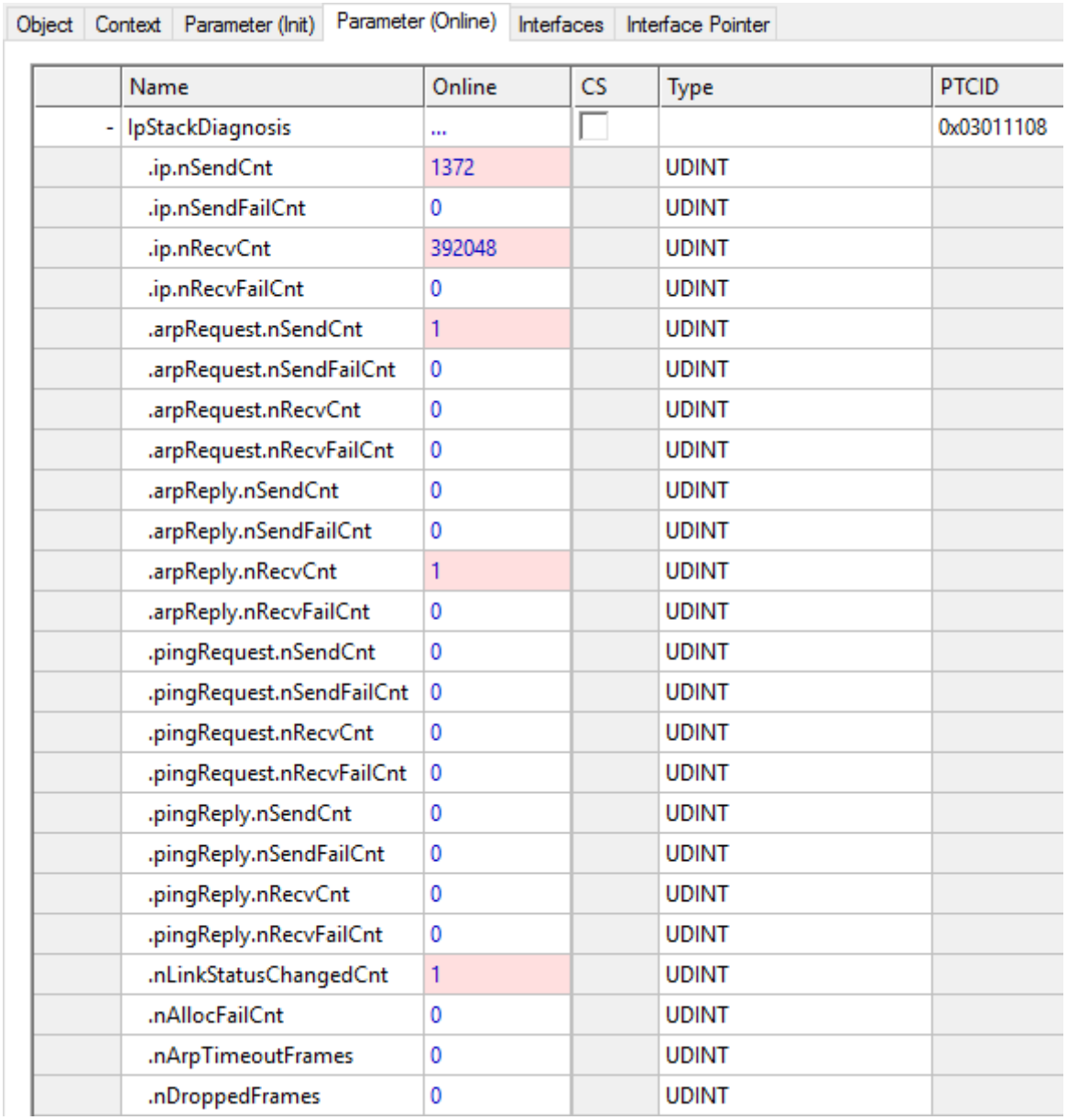


⇨ Hier sehen Sie, ob generell Frames gesendet und empfangen werden oder ob Fehler auftreten. Weiterhin können Sie an der **Recv Utilization** die Auslastung der Verbindung in Prozent ablesen. Eine sehr hohe Auslastung größer 98% deutet bereits auf eine Überlastung hin, insbesondere wenn mehrere Kameras an einem Adapter betrieben werden.

Weitere Diagnoseinformationen finden sie im IpStack.

1. Öffnen Sie dazu zuerst den unterliegenden IpStack per Doppelklick.
2. Gehen Sie zu dem **Parameter (Online)** Tab.

⇨ Hier sehen Sie die Zählerwerte zu den einzelnen Kategorien.

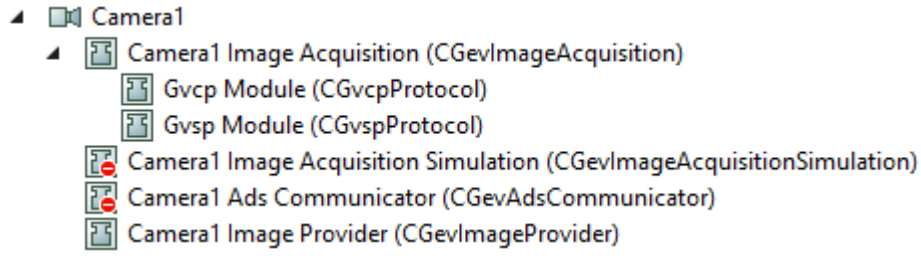


Object | Context | Parameter (Init) | Parameter (Online) | Interfaces | Interface Pointer

	Name	Online	CS	Type	PTCID
-	IpStackDiagnosis	...			0x03011108
	.ip.nSendCnt	1372		UDINT	
	.ip.nSendFailCnt	0		UDINT	
	.ip.nRecvCnt	392048		UDINT	
	.ip.nRecvFailCnt	0		UDINT	
	.arpRequest.nSendCnt	1		UDINT	
	.arpRequest.nSendFailCnt	0		UDINT	
	.arpRequest.nRecvCnt	0		UDINT	
	.arpRequest.nRecvFailCnt	0		UDINT	
	.arpReply.nSendCnt	0		UDINT	
	.arpReply.nSendFailCnt	0		UDINT	
	.arpReply.nRecvCnt	1		UDINT	
	.arpReply.nRecvFailCnt	0		UDINT	
	.pingRequest.nSendCnt	0		UDINT	
	.pingRequest.nSendFailCnt	0		UDINT	
	.pingRequest.nRecvCnt	0		UDINT	
	.pingRequest.nRecvFailCnt	0		UDINT	
	.pingReply.nSendCnt	0		UDINT	
	.pingReply.nSendFailCnt	0		UDINT	
	.pingReply.nRecvCnt	0		UDINT	
	.pingReply.nRecvFailCnt	0		UDINT	
	.nLinkStatusChangedCnt	1		UDINT	
	.nAllocFailCnt	0		UDINT	
	.nArpTimeoutFrames	0		UDINT	
	.nDroppedFrames	0		UDINT	

Diagnosemöglichkeiten in den Image Acquisition Modulen

1. Wählen Sie zuerst das jeweilige Kamera Objekt aus.



2. Öffnen Sie unter dem Gvcp Module das **Parameter (online)** Tab.

⇨ Hier erhalten Sie die Informationen zur Control Protocol Kommunikation. Das beinhaltet z. B. den Heartbeat, das Lesen und Schreiben von Registern und Kommandos wie **Start Acquisition**. Wenn alles Ok ist, sollten nur die beiden nSuccess Werte hochzählen. Wenn Timeouts auftreten, finden

Sie in diesem Timeouts [▶ 2960] Kapitel Abhilfemöglichkeiten. Bei anderen Problemen ist eine tiefergehende Diagnose mit Wireshark notwendig.

Object | Context | Parameter (Init) | Parameter (Online) | Interfaces

Name	Online	PV	CS	Unit	Type	PTCID	Comment
- GvcpProtocolDiagnosis	...					0x00010001	Contains statistics for GVCP protocol diagnosis.
.stSend.nSuccess	1242				ULINT		Number of successfully sent packets
.stSend.nError	0				ULINT		Number of packets causing errors during transmission
.stSend.nTimeout	0				ULINT		Number of timeouts that have occurred
.stRecv.nSuccess	1189				ULINT		Number of successfully received packets
.stRecv.nError	0				ULINT		Number of received packets signalling or causing errors
.stRecv.nIgnored	0				ULINT		Number of ignored packets

Die Diagnoseinformationen zum Streaming Protocol finden Sie unter dem Gvsp Module auf dem **Parameter (online)** Tab. Diese Zähler beziehen sich rein auf die übertragenen Bilddaten und sind unterteilt in die einzelnen Telegramme/Pakete und auf komplette Bilder/Blocks.

Wenn alles Ok ist, sollten nur der nSuccess und der nCompletedBlocks Wert hochzählen.

Die nResendRequests deuten bereits auf Probleme hin, konnten in diesem Fall aber noch beantwortet werden, da alle anderen Zähler noch null sind. Bei einem Timeout sollte das BlockTimeout im Configuration Assistant unter den GVSP Module Einstellungen [▶ 109] erhöht werden. Ein Hochzählen der nBlockIdOverflows ist normal und stellt keinen Fehler dar. Der Zähler läuft aufgrund des Wertebereichs irgendwann über oder wird bei einem Kameraneustart zurückgesetzt.

Object | Context | Parameter (Init) | Parameter (Online) | Interfaces | Interface Pointer

Name	Online	CS	Type	PTCID	Comment
- GvspProtocolDiagnosis	...			0x00010001	Contains statistics for GVSP protocol diagnosis.
.stRecv.nSuccess	390729		ULINT		Number of successfully received packets
.stRecv.nError	0		ULINT		Number of received packets signalling or causing errors
.stRecv.nIgnored	0		ULINT		Number of ignored packets
.nResendRequests	53		ULINT		Number of packet resend requests
.nCompletedBlocks	139		ULINT		Number of successfully received images
.nAbortedBlocks	0		ULINT		Number of aborted images
.nTimedOutBlocks	0		ULINT		Number of timed out images
.nInvalidBlocks	0		ULINT		Number of invalid images
.nFurtherErroneousBlocks	0		ULINT		Number of other erroneous images
.nDuplicateBlocks	0		ULINT		Number of duplicate images
.nMissingBlocks	0		ULINT		Number of missing images
.nUnexpectedBlocks	0		ULINT		Number of unexpected images
.nBlockIdOverflows	0		ULINT		Number of block ID overflows (not an error)

9.4 ADS Return Codes

Gruppierung der Fehlercodes:
Globale Fehlercodes: 0x0000 [▶ 2966]... (0x9811_0000 ...)
Router Fehlercodes: 0x0500 [▶ 2967]... (0x9811_0500 ...)
Allgemeine ADS Fehler: 0x0700 [▶ 2968]... (0x9811_0700 ...)
RTime Fehlercodes: 0x1000 [▶ 2970]... (0x9811_1000 ...)

Globale Fehlercodes

Hex	Dec	HRESULT	Name	Beschreibung
0x0	0	0x98110000	ERR_NOERROR	Kein Fehler.
0x1	1	0x98110001	ERR_INTERNAL	Interner Fehler.
0x2	2	0x98110002	ERR_NORTIME	Keine Echtzeit.
0x3	3	0x98110003	ERR_ALLOCLOCKEDMEM	Zuweisung gesperrt - Speicherfehler.
0x4	4	0x98110004	ERR_INSERTMAILBOX	Postfach voll – Es konnte die ADS Nachricht nicht versendet werden. Reduzieren der Anzahl der ADS Nachrichten pro Zyklus bringt Abhilfe.
0x5	5	0x98110005	ERR_WRONGRECEIVEHMSG	Falsches HMSG.
0x6	6	0x98110006	ERR_TARGETPORTNOTFOUND	Ziel-Port nicht gefunden – ADS Server ist nicht gestartet, nicht erreichbar oder nicht installiert.
0x7	7	0x98110007	ERR_TARGETMACHINENOTFOUND	Zielrechner nicht gefunden – AMS Route wurde nicht gefunden.
0x8	8	0x98110008	ERR_UNKNOWNCMDID	Unbekannte Befehl-ID.
0x9	9	0x98110009	ERR_BADTASKID	Ungültige Task-ID.
0xA	10	0x9811000A	ERR_NOIO	Kein IO.
0xB	11	0x9811000B	ERR_UNKNOWNAMSCMD	Unbekannter AMS-Befehl.
0xC	12	0x9811000C	ERR_WIN32ERROR	Win32 Fehler.
0xD	13	0x9811000D	ERR_PORTNOTCONNECTED	Port nicht verbunden.
0xE	14	0x9811000E	ERR_INVALIDAMSLENGTH	Ungültige AMS-Länge.
0xF	15	0x9811000F	ERR_INVALIDAMSNETID	Ungültige AMS Net ID.
0x10	16	0x98110010	ERR_LOWINSTLEVEL	Installations-Level ist zu niedrig –TwinCAT 2 Lizenzfehler.
0x11	17	0x98110011	ERR_NODEBUGINTAVAILABLE	Kein Debugging verfügbar.
0x12	18	0x98110012	ERR_PORTDISABLED	Port deaktiviert – TwinCAT System Service nicht gestartet.
0x13	19	0x98110013	ERR_PORTALREADYCONNECTED	Port bereits verbunden.
0x14	20	0x98110014	ERR_AMSSYNC_W32ERROR	AMS Sync Win32 Fehler.
0x15	21	0x98110015	ERR_AMSSYNC_TIMEOUT	AMS Sync Timeout.
0x16	22	0x98110016	ERR_AMSSYNC_AMSERROR	AMS Sync Fehler.
0x17	23	0x98110017	ERR_AMSSYNC_NOINDEXINMAP	Keine Index-Map für AMS Sync vorhanden.
0x18	24	0x98110018	ERR_INVALIDAMSPORT	Ungültiger AMS-Port.
0x19	25	0x98110019	ERR_NOMEMORY	Kein Speicher.
0x1A	26	0x9811001A	ERR_TCPSEND	TCP Sendefehler.
0x1B	27	0x9811001B	ERR_HOSTUNREACHABLE	Host nicht erreichbar.
0x1C	28	0x9811001C	ERR_INVALIDAMSFRAGMENT	Ungültiges AMS Fragment.
0x1D	29	0x9811001D	ERR_TLSSEND	TLS Sendefehler – Secure ADS Verbindung fehlgeschlagen.
0x1E	30	0x9811001E	ERR_ACCESSDENIED	Zugriff Verweigert – Secure ADS Zugriff verweigert.

Router Fehlercodes

Hex	Dec	HRESULT	Name	Beschreibung
0x500	1280	0x98110500	ROUTERERR_NOLOCKEDMEMORY	Lockierter Speicher kann nicht zugewiesen werden.
0x501	1281	0x98110501	ROUTERERR_RESIZEMEMORY	Die Größe des Routerspeichers konnte nicht geändert werden.
0x502	1282	0x98110502	ROUTERERR_MAILBOXFULL	Das Postfach hat die maximale Anzahl der möglichen Meldungen erreicht.
0x503	1283	0x98110503	ROUTERERR_DEBUGBOXFULL	Das Debug Postfach hat die maximale Anzahl der möglichen Meldungen erreicht.
0x504	1284	0x98110504	ROUTERERR_UNKNOWNPORTTYPE	Der Porttyp ist unbekannt.
0x505	1285	0x98110505	ROUTERERR_NOTINITIALIZED	Router ist nicht initialisiert.
0x506	1286	0x98110506	ROUTERERR_PORTALREADYINUSE	Die Portnummer ist bereits vergeben.
0x507	1287	0x98110507	ROUTERERR_NOTREGISTERED	Der Port ist nicht registriert.
0x508	1288	0x98110508	ROUTERERR_NOMOREQUEUES	Die maximale Portanzahl ist erreicht.
0x509	1289	0x98110509	ROUTERERR_INVALIDPORT	Der Port ist ungültig.
0x50A	1290	0x9811050A	ROUTERERR_NOTACTIVATED	Der Router ist nicht aktiv.
0x50B	1291	0x9811050B	ROUTERERR_FRAGMENTBOXFULL	Das Postfach hat die maximale Anzahl für fragmentierte Nachrichten erreicht.
0x50C	1292	0x9811050C	ROUTERERR_FRAGMENTTIMEOUT	Fragment Timeout aufgetreten.
0x50D	1293	0x9811050D	ROUTERERR_TOBEREMOVED	Port wird entfernt.

Allgemeine ADS Fehlercodes

Hex	Dec	HRESULT	Name	Beschreibung
0x700	1792	0x98110700	ADSERR_DEVICE_ERROR	Allgemeiner Gerätefehler.
0x701	1793	0x98110701	ADSERR_DEVICE_SRVNOTSUPP	Service wird vom Server nicht unterstützt.
0x702	1794	0x98110702	ADSERR_DEVICE_INVALIDGRP	Ungültige Index-Gruppe.
0x703	1795	0x98110703	ADSERR_DEVICE_INVALIDOFFSET	Ungültiger Index-Offset.
0x704	1796	0x98110704	ADSERR_DEVICE_INVALIDACCESS	Lesen oder Schreiben nicht gestattet. Mehrere Ursachen sind möglich. Beispielsweise beim Anlegen von Routen, dass ein falsches Passwort angegeben wurde.
0x705	1797	0x98110705	ADSERR_DEVICE_INVALIDSIZE	Parametergröße nicht korrekt.
0x706	1798	0x98110706	ADSERR_DEVICE_INVALIDDATA	Ungültige Daten-Werte.
0x707	1799	0x98110707	ADSERR_DEVICE_NOTREADY	Gerät nicht betriebsbereit.
0x708	1800	0x98110708	ADSERR_DEVICE_BUSY	Gerät beschäftigt.
0x709	1801	0x98110709	ADSERR_DEVICE_INVALIDCONTEXT	Ungültiger Kontext vom Betriebssystem - Kann durch Verwendung von ADS Bausteinen in unterschiedlichen Tasks auftreten. Abhilfe kann die Multitasking-Syncronisation in der SPS geben.
0x70A	1802	0x9811070A	ADSERR_DEVICE_NOMEMORY	Nicht genügend Speicher.
0x70B	1803	0x9811070B	ADSERR_DEVICE_INVALIDPARM	Ungültige Parameter-Werte.
0x70C	1804	0x9811070C	ADSERR_DEVICE_NOTFOUND	Nicht gefunden (Dateien,...).
0x70D	1805	0x9811070D	ADSERR_DEVICE_SYNTAX	Syntax-Fehler in Datei oder Befehl.
0x70E	1806	0x9811070E	ADSERR_DEVICE_INCOMPATIBLE	Objekte stimmen nicht überein.
0x70F	1807	0x9811070F	ADSERR_DEVICE_EXISTS	Objekt ist bereits vorhanden.
0x710	1808	0x98110710	ADSERR_DEVICE_SYMBOLNOTFOUND	Symbol nicht gefunden.
0x711	1809	0x98110711	ADSERR_DEVICE_SYMBOLVERSIONINVALID	Symbol-Version ungültig – Kann durch einen Online-Change auftreten. Erzeuge einen neuen Handle.
0x712	1810	0x98110712	ADSERR_DEVICE_INVALIDSTATE	Gerät (Server) ist im ungültigen Zustand.
0x713	1811	0x98110713	ADSERR_DEVICE_TRANSMODENOTSUPP	AdsTransMode nicht unterstützt.
0x714	1812	0x98110714	ADSERR_DEVICE_NOTIFYHNDINVALID	Notification Handle ist ungültig.
0x715	1813	0x98110715	ADSERR_DEVICE_CLIENTUNKNOWN	Notification-Client nicht registriert.
0x716	1814	0x98110716	ADSERR_DEVICE_NOMOREHDLS	Keine weiteren Handles verfügbar.
0x717	1815	0x98110717	ADSERR_DEVICE_INVALIDWATCHSIZE	Größe der Notification zu groß.
0x718	1816	0x98110718	ADSERR_DEVICE_NOTINIT	Gerät nicht initialisiert.
0x719	1817	0x98110719	ADSERR_DEVICE_TIMEOUT	Gerät hat einen Timeout.
0x71A	1818	0x9811071A	ADSERR_DEVICE_NOINTERFACE	Interface Abfrage fehlgeschlagen.
0x71B	1819	0x9811071B	ADSERR_DEVICE_INVALIDINTERFACE	Falsches Interface angefordert.
0x71C	1820	0x9811071C	ADSERR_DEVICE_INVALIDCLSID	Class-ID ist ungültig.
0x71D	1821	0x9811071D	ADSERR_DEVICE_INVALIDOBJID	Object-ID ist ungültig.
0x71E	1822	0x9811071E	ADSERR_DEVICE_PENDING	Anforderung steht aus.
0x71F	1823	0x9811071F	ADSERR_DEVICE_ABORTED	Anforderung wird abgebrochen.
0x720	1824	0x98110720	ADSERR_DEVICE_WARNING	Signal-Warnung.
0x721	1825	0x98110721	ADSERR_DEVICE_INVALIDARRAYIDX	Ungültiger Array-Index.
0x722	1826	0x98110722	ADSERR_DEVICE_SYMBOLNOTACTIVE	Symbol nicht aktiv.
0x723	1827	0x98110723	ADSERR_DEVICE_ACCESSDENIED	Zugriff verweigert. Mehrere Ursachen sind möglich. Beispielsweise, dass eine Unidirectionale ADS Route in die umgekehrte Richtung verwendet wird.
0x724	1828	0x98110724	ADSERR_DEVICE_LICENSENOTFOUND	Fehlende Lizenz.
0x725	1829	0x98110725	ADSERR_DEVICE_LICENSEEXPIRED	Lizenz abgelaufen.
0x726	1830	0x98110726	ADSERR_DEVICE_LICENSEEXCEEDED	Lizenz überschritten.
0x727	1831	0x98110727	ADSERR_DEVICE_LICENSEINVALID	Lizenz ungültig.
0x728	1832	0x98110728	ADSERR_DEVICE_LICENSESYSTEMID	Lizenzproblem: System-ID ist ungültig.
0x729	1833	0x98110729	ADSERR_DEVICE_LICENSENOTIMELIMIT	Lizenz nicht zeitlich begrenzt.
0x72A	1834	0x9811072A	ADSERR_DEVICE_LICENSEFUTUREISSUE	Lizenzproblem: Zeitpunkt in der Zukunft.
0x72B	1835	0x9811072B	ADSERR_DEVICE_LICENSETIMETOLONG	Lizenz-Zeitraum zu lang.
0x72C	1836	0x9811072C	ADSERR_DEVICE_EXCEPTION	Exception beim Systemstart.
0x72D	1837	0x9811072D	ADSERR_DEVICE_LICENSEDUPLICATED	Lizenz-Datei zweimal gelesen.
0x72E	1838	0x9811072E	ADSERR_DEVICE_SIGNATUREINVALID	Ungültige Signatur.
0x72F	1839	0x9811072F	ADSERR_DEVICE_CERTIFICATEINVALID	Zertifikat ungültig.
0x730	1840	0x98110730	ADSERR_DEVICE_LICENSEOEMNOTFOUND	Public Key vom OEM nicht bekannt.
0x731	1841	0x98110731	ADSERR_DEVICE_LICENSERESTRICTED	Lizenz nicht gültig für diese System.ID.

Hex	Dec	HRESULT	Name	Beschreibung
0x732	1842	0x98110732	ADSERR_DEVICE_LICENSEDEMODENIED	Demo-Lizenz untersagt.
0x733	1843	0x98110733	ADSERR_DEVICE_INVALIDFNCID	Funktions-ID ungültig.
0x734	1844	0x98110734	ADSERR_DEVICE_OUTOFRANGE	Außerhalb des gültigen Bereiches.
0x735	1845	0x98110735	ADSERR_DEVICE_INVALIDALIGNMENT	Ungültiges Alignment.
0x736	1846	0x98110736	ADSERR_DEVICE_LICENSEPLATFORM	Ungültiger Plattform Level.
0x737	1847	0x98110737	ADSERR_DEVICE_FORWARD_PL	Kontext – Weiterleitung zum Passiv-Level.
0x738	1848	0x98110738	ADSERR_DEVICE_FORWARD_DL	Kontext – Weiterleitung zum Dispatch-Level.
0x739	1849	0x98110739	ADSERR_DEVICE_FORWARD_RT	Kontext – Weiterleitung zur Echtzeit.
0x740	1856	0x98110740	ADSERR_CLIENT_ERROR	Clientfehler.
0x741	1857	0x98110741	ADSERR_CLIENT_INVALIDPARM	Dienst enthält einen ungültigen Parameter.
0x742	1858	0x98110742	ADSERR_CLIENT_LISTEMPTY	Polling-Liste ist leer.
0x743	1859	0x98110743	ADSERR_CLIENT_VARUSED	Var-Verbindung bereits im Einsatz.
0x744	1860	0x98110744	ADSERR_CLIENT_DUPLINVOKEID	Die aufgerufene ID ist bereits in Benutzung.
0x745	1861	0x98110745	ADSERR_CLIENT_SYNCTIMEOUT	Timeout ist aufgetreten – Die Gegenstelle antwortet nicht im vorgegebenen ADS Timeout. Die Routeneinstellung der Gegenstelle kann falsch konfiguriert sein.
0x746	1862	0x98110746	ADSERR_CLIENT_W32ERROR	Fehler im Win32 Subsystem.
0x747	1863	0x98110747	ADSERR_CLIENT_TIMEOUTINVALID	Ungültiger Client Timeout-Wert.
0x748	1864	0x98110748	ADSERR_CLIENT_PORTNOTOPEN	Port nicht geöffnet.
0x749	1865	0x98110749	ADSERR_CLIENT_NOAMSADDR	Keine AMS Adresse.
0x750	1872	0x98110750	ADSERR_CLIENT_SYNCINTERNAL	Interner Fehler in Ads-Sync.
0x751	1873	0x98110751	ADSERR_CLIENT_ADDHASH	Überlauf der Hash-Tabelle.
0x752	1874	0x98110752	ADSERR_CLIENT_REMOVEHASH	Schlüssel in der Tabelle nicht gefunden.
0x753	1875	0x98110753	ADSERR_CLIENT_NOMORESYM	Keine Symbole im Cache.
0x754	1876	0x98110754	ADSERR_CLIENT_SYNCRESINVALID	Ungültige Antwort erhalten.
0x755	1877	0x98110755	ADSERR_CLIENT_SYNCPORTLOCKED	Sync Port ist verriegelt.
0x756	1878	0x98110756	ADSERR_CLIENT_REQUESTCANCELLED	Die Anfrage wurde abgebrochen.

RTime Fehlercodes

Hex	Dec	HRESULT	Name	Beschreibung
0x1000	4096	0x98111000	RTERR_INTERNAL	Interner Fehler im Echtzeit-System.
0x1001	4097	0x98111001	RTERR_BADTIMERPERIODS	Timer-Wert nicht gültig.
0x1002	4098	0x98111002	RTERR_INVALIDTASKPTR	Task-Pointer hat den ungültigen Wert 0 (null).
0x1003	4099	0x98111003	RTERR_INVALIDSTACKPTR	Stack-Pointer hat den ungültigen Wert 0 (null).
0x1004	4100	0x98111004	RTERR_PRIOEXISTS	Die Request Task Priority ist bereits vergeben.
0x1005	4101	0x98111005	RTERR_NOMORETCB	Kein freier TCB (Task Control Block) verfügbar. Maximale Anzahl von TCBs beträgt 64.
0x1006	4102	0x98111006	RTERR_NOMORESEMAS	Keine freien Semaphoren zur Verfügung. Maximale Anzahl der Semaphoren beträgt 64.
0x1007	4103	0x98111007	RTERR_NOMOREQUEUES	Kein freier Platz in der Warteschlange zur Verfügung. Maximale Anzahl der Plätze in der Warteschlange beträgt 64.
0x100D	4109	0x9811100D	RTERR_EXTIRQALREADYDEF	Ein externer Synchronisations-Interrupt wird bereits angewandt.
0x100E	4110	0x9811100E	RTERR_EXTIRQNOTDEF	Kein externer Sync-Interrupt angewandt.
0x100F	4111	0x9811100F	RTERR_EXTIRQINSTALLFAILED	Anwendung des externen Synchronisierungs-Interrupts ist fehlgeschlagen.
0x1010	4112	0x98111010	RTERR_IRQLNOTLESSOREQUAL	Aufruf einer Service-Funktion im falschen Kontext
0x1017	4119	0x98111017	RTERR_VMXNOTSUPPORTED	Intel VT-x Erweiterung wird nicht unterstützt.
0x1018	4120	0x98111018	RTERR_VMXDISABLED	Intel VT-x Erweiterung ist nicht aktiviert im BIOS.
0x1019	4121	0x98111019	RTERR_VMXCONTROLSMISSING	Fehlende Funktion in Intel VT-x Erweiterung.
0x101A	4122	0x9811101A	RTERR_VMXENABLEFAILS	Aktivieren von Intel VT-x schlägt fehl.

Spezifische positive HRESULT Return Codes:

HRESULT	Name	Beschreibung
0x0000_0000	S_OK	Kein Fehler.
0x0000_0001	S_FALSE	Kein Fehler. Bsp.: erfolgreiche Abarbeitung, bei der jedoch ein negatives oder unvollständiges Ergebnis erzielt wurde.
0x0000_0203	S_PENDING	Kein Fehler. Bsp.: erfolgreiche Abarbeitung, bei der jedoch noch kein Ergebnis vorliegt.
0x0000_0256	S_WATCHDOG_TIMEOUT	Kein Fehler. Bsp.: erfolgreiche Abarbeitung, bei der jedoch eine Zeitüberschreitung eintrat.

TCP Winsock-Fehlercodes

Hex	Dec	Name	Beschreibung
0x274C	10060	WSAETIMEDOUT	Verbindungs Timeout aufgetreten - Fehler beim Herstellen der Verbindung, da die Gegenstelle nach einer bestimmten Zeitspanne nicht ordnungsgemäß reagiert hat, oder die hergestellte Verbindung konnte nicht aufrecht erhalten werden, da der verbundene Host nicht reagiert hat.
0x274D	10061	WSAECONNREFUSED	Verbindung abgelehnt - Es konnte keine Verbindung hergestellt werden, da der Zielcomputer dies explizit abgelehnt hat. Dieser Fehler resultiert normalerweise aus dem Versuch, eine Verbindung mit einem Dienst herzustellen, der auf dem fremden Host inaktiv ist—das heißt, einem Dienst, für den keine Serveranwendung ausgeführt wird.
0x2751	10065	WSAEHOSTUNREACH	Keine Route zum Host - Ein Socketvorgang bezog sich auf einen nicht verfügbaren Host.
Weitere Winsock-Fehlercodes: Win32-Fehlercodes			

9.5 Support und Service

Beckhoff und seine weltweiten Partnerfirmen bieten einen umfassenden Support und Service, der eine schnelle und kompetente Unterstützung bei allen Fragen zu Beckhoff Produkten und Systemlösungen zur Verfügung stellt.

Downloadfinder

Unser Downloadfinder beinhaltet alle Dateien, die wir Ihnen zum Herunterladen anbieten. Sie finden dort Applikationsberichte, technische Dokumentationen, technische Zeichnungen, Konfigurationsdateien und vieles mehr.

Die Downloads sind in verschiedenen Formaten erhältlich.

Beckhoff Niederlassungen und Vertretungen

Wenden Sie sich bitte an Ihre Beckhoff Niederlassung oder Ihre Vertretung für den lokalen Support und Service zu Beckhoff Produkten!

Die Adressen der weltweiten Beckhoff Niederlassungen und Vertretungen entnehmen Sie bitte unserer Internetseite: www.beckhoff.com

Dort finden Sie auch weitere Dokumentationen zu Beckhoff Komponenten.

Beckhoff Support

Der Support bietet Ihnen einen umfangreichen technischen Support, der Sie nicht nur bei dem Einsatz einzelner Beckhoff Produkte, sondern auch bei weiteren umfassenden Dienstleistungen unterstützt:

- Support
- Planung, Programmierung und Inbetriebnahme komplexer Automatisierungssysteme
- umfangreiches Schulungsprogramm für Beckhoff Systemkomponenten

Hotline:	+49 5246 963-157
E-Mail:	support@beckhoff.com

Beckhoff Service

Das Beckhoff Service-Center unterstützt Sie rund um den After-Sales-Service:

- Vor-Ort-Service
- Reparaturservice
- Ersatzteilservice
- Hotline-Service

Hotline:	+49 5246 963-460
E-Mail:	service@beckhoff.com

Beckhoff Unternehmenszentrale

Beckhoff Automation GmbH & Co. KG

Hülshorstweg 20
33415 Verl
Deutschland

Telefon:	+49 5246 963-0
E-Mail:	info@beckhoff.com
Internet:	www.beckhoff.com

9.6 Third-party Components

Diese Software enthält Komponenten von Drittanbietern. Weitere Informationen entnehmen Sie bitte der Lizenzdatei, die in folgenden Ordnern bereitgestellt wird:

Unterschiedliche Speicherpfade in TwinCAT 3.1 4024 und TwinCAT 3.1 4026

Bis einschließlich Build 4024:

<TwinCatInstallPath>\3.1\System\Legal\TF7xxx Additional license information.txt

Ab Build 4026:

- *C:\Program Files (x86)\Beckhoff\Legal\TwinCAT-XAE-GigEVision*
- *C:\Program Files (x86)\Beckhoff\Legal\TwinCAT-XAE-TMX-Vision*
- *C:\Program Files (x86)\Beckhoff\Legal\TwinCAT-XAE-Vision*
- *C:\Program Files (x86)\Beckhoff\Legal\TwinCAT-XAR-VisionService*

Trademark statements

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered trademarks of and licensed by Beckhoff Automation GmbH.

Third-party trademark statements

Chrome, Chromium and Google are trademarks of Google LLC.

DSP System Toolbox, Embedded Coder, MATLAB, MATLAB Coder, MATLAB Compiler, MathWorks, Predictive Maintenance Toolbox, Simscape, Simscape™ Multibody™, Simulink, Simulink Coder, Stateflow and ThingSpeak are registered trademarks of The MathWorks, Inc.

Intel, the Intel logo, Intel Core, Xeon, Intel Atom, Celeron and Pentium are trademarks of Intel Corporation or its subsidiaries.

Microsoft, Microsoft Azure, Microsoft Edge, PowerShell, Visual Studio, Windows and Xbox are trademarks of the Microsoft group of companies.

Mozilla and Firefox are trademarks of the Mozilla Foundation in the U.S. and other countries.

Wireshark is a registered trademark of Sysdig, Inc.

Mehr Informationen:
www.beckhoff.com/twincat-vision/

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Deutschland
Telefon: +49 5246 9630
info@beckhoff.com
www.beckhoff.com